	Page 1
1	UNITED STATES PATENT AND TRADEMARK OFFICE
2	BEFORE THE PATENT TRIAL AND APPEAL BOARD
3	
4	
)
5	LG ELECTRONICS, INC.,)
)
6	Petitioner,)
)
7	vs.) Nos. IPR2015-00326
) IPR2015-00330
8	ATI TECHNOLOGIES ULC,)
)
9	Patent Owner.)
)
10	
11	
12	
13	
14	
15	VIDEOTAPED DEPOSITION OF NADER BAGHERZADEH, Ph.D.
16	Los Angeles, California
17	Tuesday, September 15, 2015
18	Volume I
19	
2021	Manihart Land Calutions
Z T	Veritext Legal Solutions Mid-Atlantic Region
	1250 Eye Street NW - Suite 1201
22	Washington, D.C. 20005
23	wasiiiigcoii, D.C. 20005
24	
25	
ر ہے	

Veritext Legal Solutions 215-241-1000 ~ 610-434-8588 ~ 302-571-0510 ~ 202-803-8830

ATI 2073 LG v. ATI IPR2015-00326

```
Page 4
     UNITED STATES PATENT AND TRADEMARK OFFICE
                                                     1 APPEARANCES (Continued):
 2
     BEFORE THE PATENT TRIAL AND APPEAL BOARD
                                                    2
 3
                                                    3
                                                      For Patent Owner:
 4
                                                    4
                                                          STERNE KESSLER GOLDSTEIN FOX
5 LG ELECTRONICS, INC.,
                                                    5
                                                          BY: JONATHAN TUMINARO, Ph.D.
                                                    6
                                                          BY: TYLER J. DUTTON
       Petitioner.
 6
                                                    7
                                                          Attorneys at Law
                                                    8
                                                          1100 New York Avenue, NW
 7
                ) Nos. IPR2015-00326
                                                    9
                   IPR2015-00330
                                                          Washington, D.C. 20005
   ATI TECHNOLOGIES ULC,
                                                   10
                                                          202-371-2600
                                                   11
                                                          jtuminar@skgf.com
       Patent Owner.
                                                   12
                                                          tdutton@skgf.com
                                                   13
10
11
                                                   14
                                                       Videographer:
12
                                                   15
                                                          GRANT CIHLAR
13
                                                   16
14
      Videotaped deposition of NADER BAGHERZADEH,
                                                   17
15 Ph.D., Volume I, taken on behalf of Patent Owner, at
16 350 South Grand Avenue, Suite 2500, Los Angeles,
                                                   18
17 California, beginning at 9:38 a.m. and ending at
                                                   19
   1:42 p.m. on Tuesday, September 15, 2015, before
                                                   20
19 NADIA NEWHART, Certified Shorthand Reporter
                                                   21
20 No. 8714.
21
                                                   22
22
                                                   23
23
                                                   24
24
                                                   25
25
                                             Page 3
                                                                                                Page 5
 1 APPEARANCES:
                                                                    INDEX
                                                    1
 2
                                                    2 WITNESS
                                                                                   EXAMINATION
 3
    For Petitioner:
                                                    3 NADER BAGHERZADEH, Ph.D.
 4
       MAYER BROWN, LLP
                                                       Volume I
 5
       BY: ROBERT G. PLUTA
                                                    5
                                                                BY MR. TUMINARO
                                                                                              8
 6
       Attorney at Law
                                                    6
                                                                BY MR. PLUTA
                                                                                          123
 7
       71 South Wacker Drive
                                                    7
 8
       Chicago, Illinois 60606-4637
                                                    8
                                                                   EXHIBITS
 9
       312-701-8641
                                                    9
                                                      NUMBER
                                                                         DESCRIPTION
                                                                                               PAGE
10
       rpluta@mayerbrown.com
                                                      Exhibit 1 Declaration of Dr. Nader
                                                                                              10
11
                                                   11
                                                              Bagherzadeh; 111 pages
12
       MAYER BROWN, LLP
                                                   12
13
       BY: JOHN X. ZHU
                                                   13
                                                       Exhibit 2 Declaration of Dr. Nader
                                                                                              11
14
       Attorney at Law
                                                   14
                                                              Bagherzadeh; 86 pages
15
       1999 K Street, N.W.
                                                   15
       Washington, D.C. 20006-1101
16
                                                   16 Exhibit 3 Consultant Curriculum Vitae
                                                                                                15
17
       202-263-3318
                                                   17
                                                              for Nader Bagherzadeh, Ph.D.;
18
       jzhu@mayerbrown.com
                                                   18
                                                              33 pages
19
                                                   19
20
                                                   20
                                                      Exhibit 4 Article entitled "How GPUs
                                                                                               36
21
                                                   21
                                                              Work"; 5 pages
22
                                                   22
23
                                                   23 Exhibit 5 Article entitled "Exploiting
                                                                                              40
24
                                                   24
                                                              the Shader Model 4.0
25
                                                   25
                                                              Architecture"; 9 pages
```

2 (Pages 2 - 5)

1	INDEX (Continued):	Page 6	1	Page 8 MR. TUMINARO: Jonathan Tuminaro from the law	
2	EXHIBITS			firm of Sterne Kessler Goldstein & Fox on behalf of	
3	NUMBER DESCRIPTION	PAGE	3		
4	Exhibit 6 United States Patent	61		4 is Tyler Dutton, also from Sterne Kessler.	
5	Number 5,808,690; 50 pages	01	5	MR. PLUTA: Robert Pluta from Mayer Brown on	
6	1 tamoer 5,000,000, 50 pages		6		
7	Exhibit 7 United States Patent	61	7	MR. ZHU: John Zhu also of Mayer Brown for LG	
8	Number 6,897,871; 10 pages		8	Electronics.	
9	,		9	THE VIDEOGRAPHER: Thank you.	
10	Exhibit 8 United States Patent	84	10	The witness will be sworn in, and counsel may	
11	Number 7,015,913; 20 pages		11	begin the examination.	
12	, , , , 1 &		12		
13	Exhibit 9 United States Patent	98	13	NADER BAGHERZADEH, Ph.D.,	
14	Number 7,376,811; 12 pages		14	having been first duly sworn, was examined and	
15	, , , , 1 &		15	testified as follows:	
16	Exhibit 10 Patent Trial and Appeal	104	16		
17	Board Decision; 28 pages		17	EXAMINATION	
18			18	BY MR. TUMINARO:	
19			19	Q Good morning, sir.	
20			20	A Good morning.	
21			21	Q This is the second time I'm taking your	
22			22	deposition, right?	
23			23	A Correct.	
24			24	Q Correct. Okay. Just a couple ground rules	
25			25	for this deposition as we did last time. We're	
		Page 7		Page 9	
1	Los Angeles, California, Tuesday, September	er 15, 2015	1	trying to get a clear record from the court	
2	9:38 a.m.		2	reporter, so I'll ask that you don't speak over me	
3			3	and I'll try not to speak over you; is that fair?	
4	THE VIDEOGRAPHER: Good morning	g. We're on	4	A That's good.	
5	the record. The time is 9:38 a.m. on		5	Q Okay. I'm going to try to ask clear	
6	September 15th, 2015. This is the video-rec	orded	6	questions, but if at any time you don't understand	
7	deposition of Dr. Nader Bagherzadeh.			questions, but it at any time you don't understand	
			7		
8	My name is Grant Cihlar, here with our	court	7 8	my question, will you let me know? A Sure.	
9	reporter, Nadia Newhart. We are here from	Veritext	8	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm	
9	•	Veritext	8	my question, will you let me know? A Sure.	
9	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for	Veritext the	8 9 10 11	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair.	
9 10 11 12	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California.	Veritext the at	8 9 10 11 12	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but	
9 10 11	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic	Veritext the at	8 9 10 11 12 13	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me	
9 10 11 12 13 14	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC	Veritext the at cs, The case	8 9 10 11 12 13 14	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know?	
9 10 11 12 13 14 15	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0	Veritext the at cs, The case 0326.	8 9 10 11 12 13 14 15	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes.	
9 10 11 12 13 14 15 16	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recording.	Veritext the at cs, The case 0326.	8 9 10 11 12 13 14 15 16	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's	
9 10 11 12 13 14 15 16	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordic will take place unless all parties agree to go	Veritext the at cs, The case 0326. ng	8 9 10 11 12 13 14 15 16 17	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the	
9 10 11 12 13 14 15 16 17 18	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordi will take place unless all parties agree to go the record. Microphones are sensitive and many control of the record.	Veritext the at cs, The case 0326. ng off aay pick	8 9 10 11 12 13 14 15 16 17 18	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break;	
9 10 11 12 13 14 15 16 17 18 19	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordic will take place unless all parties agree to go the record. Microphones are sensitive and mup whispers, private conversations and cellured.	Veritext the at CS, The case 0326. ng off nay pick lar	8 9 10 11 12 13 14 15 16 17 18	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break; is that fair?	
9 10 11 12 13 14 15 16 17 18 19 20	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordic will take place unless all parties agree to go the record. Microphones are sensitive and mup whispers, private conversations and celluinterference. I am a notary public. I am not	Veritext the at CS, The case 0326. ng off nay pick lar	8 9 10 11 12 13 14 15 16 17 18 19 20	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break; is that fair? A That is fair.	
9 10 11 12 13 14 15 16 17 18 19 20 21	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordic will take place unless all parties agree to go the record. Microphones are sensitive and mup whispers, private conversations and celluinterference. I am a notary public. I am not related to any party in this action, nor am I	Veritext the at cs, The case 0326. ng off nay pick lar	8 9 10 11 12 13 14 15 16 17 18 19 20 21	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break; is that fair? A That is fair. Q Okay. You understand that you are testifying	
9 10 11 12 13 14 15 16 17 18 19 20 21 22	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordic will take place unless all parties agree to gothe record. Microphones are sensitive and mup whispers, private conversations and celluinterference. I am a notary public. I am not related to any party in this action, nor am I financially interested in the outcome in any sensitive and mup whose sensitive and mup whispers, private conversations and celluinterference. I am a notary public. I am not related to any party in this action, nor am I financially interested in the outcome in any sensitive and mup whose sensitive and mup wh	Veritext the at at as, The case 0326. ng off nay pick lar	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break; is that fair? A That is fair. Q Okay. You understand that you are testifying under oath here today?	
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordi will take place unless all parties agree to go the record. Microphones are sensitive and mup whispers, private conversations and cellu interference. I am a notary public. I am not related to any party in this action, nor am I financially interested in the outcome in any of there are any objections to proceeding, please.	Veritext the at at as, The case 0326. ag off aay pick lar way. ase	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break; is that fair? A That is fair. Q Okay. You understand that you are testifying under oath here today? A Correct.	
9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	reporter, Nadia Newhart. We are here from Legal Solutions at the request of counsel for patent owner. This deposition is being held Mayer Brown in Los Angeles, California. The caption of this case is LG Electronic Incorporated versus ATI Technologies ULC numbers are IPR2015-00330 and IPR2015-0 Please note that audio and video recordic will take place unless all parties agree to gothe record. Microphones are sensitive and mup whispers, private conversations and celluinterference. I am a notary public. I am not related to any party in this action, nor am I financially interested in the outcome in any sensitive and mup whose sensitive and mup whispers, private conversations and celluinterference. I am a notary public. I am not related to any party in this action, nor am I financially interested in the outcome in any sensitive and mup whose sensitive and mup wh	Veritext the at at as, The case 0326. ag off aay pick lar way. ase	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	my question, will you let me know? A Sure. Q Okay. If you answer one of my questions, I'm going to assume you understood it; is that fair? A That's fair. Q Okay. I'm going to take periodic breaks, but if at any time you need a break, will you let me know? A Yes. Q Okay. One thing I'd ask, though, if there's a pending question, I ask that you answer the que answer the question before we take a break; is that fair? A That is fair. Q Okay. You understand that you are testifying under oath here today?	

3 (Pages 6 - 9)

Page 10 Q Are there any changes at all that you would Q Okay. You submitted two declarations in 1 1 2 these two cases? 2 like to make to your declaration in the 326 IPR? A Two cases, two declarations, yes. 3 A Nothing comes to mind right now. MR. TUMINARO: Okay. I'd like to have this Q Okay. Same with respect to Exhibit 2, which 4 5 marked. 5 is your declaration in the 330 IPR. Anything that (Exhibit 1 was marked for identification you would like to add to that declaration? 7 by the court reporter and is attached hereto.) 8 BY MR. TUMINARO: 8 Q Anything that you'd like to delete? Q Sir, you've been handed what's been marked as 9 A No. 10 Exhibit 1 for identification purposes. This is the 10 Q Any changes at all that you'd like to make to declaration that you submitted in the 326 IPR; is 11 that declaration? 12 that right? 12 A Not at this time. A Correct. 13 13 Q Okay. When were you first contacted with 14 Q And if you would turn with me to the last 14 respect to these two IPRs? 15 page before Appendix A. 15 A So this was signed December. I would say --16 A Exhibit 1? 16 I'm guessing early fall. 17 O Exhibit 1. 17 O 2014? 18 A Last page? 18 A Yeah. 19 Q Before Appendix A. 19 Q Okay. And who contacted you? 20 20 A I think it was Mr. Maas and Mr. Zhu and A Yes. 21 others from that team, I think. 21 Q Is that your signature that appears on that 22 last page that's not numbered? 22 Q Okay. 23 23 A Correct. A I'm -- I'm not certain about this, but these 24 Q Okay. And you signed this declaration on are the names that come to mind. 25 December 10th, 2014? 25 Q That's your best recollection? Page 11 Page 13 1 A Correct. 1 A That's right. 2 MR. TUMINARO: Okay. Q Okay. And how were you contacted? By 3 (Exhibit 2 was marked for identification 3 telephone? By e-mail? In what form were you by the court reporter and is attached hereto.) contacted? 5 BY MR. TUMINARO: 5 A First time? Q If you turn with me to Exhibit 2, this is the 6 O Yes. 7 7 declaration that you submitted in 330 IPR; is that A Phone call, I would say, yeah. 8 right? Q Okay. And since the time that you were first 9 A Correct. contacted until you submitted your declaration, how Q Okay. And again, if you would just turn with 10 much time did you spend preparing your declaration? me to the last page before the appendix. Is that A Oh, I'm going to give you a lower bound like 12 your signature? 12 I have done in the past for my depositions. It was 13 A Yes. 13 at least 40 hours. 14 Q Okay. And you signed the declaration in the Q At least 40 hours on each or combined? Each 15 330 IPR on December 9th, 2014; is that right? 15 declaration --

A That is correct. 16

Q Okay. All right. With respect to Exhibit 1, 17

18 is there anything that you would like to add?

19 MR. PLUTA: Object to the form.

20 THE WITNESS: Not that I can think of right

21 now.

22 BY MR. TUMINARO:

Q Is there anything that you would like to 23

24 delete?

25 A Not that I can think of right now.

16 A On each.

17 O On each.

18 So a total of at least 80 hours working on

19 your declaration in the 326 --

20 A And the --

21 Q -- and the 330?

22 A I'm sorry. Because there was some other

23 case, as well, so that's -- there were -- sorry.

24 There were other patents involved, so I would say --

25 I would say 40 hours each of these two, no. I would

4 (Pages 10 - 13)

Page 14 Page 16

- 1 say total, about 40 hours or more.
- 2 Q Okay.
- 3 A Yeah, I think that's correct.
- 4 Q Okay. And what did you do to prepare for
- 5 today's deposition?
- A I reviewed my declaration. I looked at all
- 7 the appendix references. I can -- I looked at the
- patent. I glanced over the patent history. I've
- 9 looked at Lindholm's patent, Rich Kizhepat,
- 10 Kurihara. I looked at certain portions of OpenGL
- 11 that I was interested in, and that's it.
- 12 Q Okay. And did you also -- so with respect --
- 13 strike that.
- 14 You looked at the 871 patent; is that right?
- 15 A Yes.
- Q Did you also look at the 369 patent? 16
- 17 A Yes, I did.
- 18 Q Okay. Did you also look at the prosecution
- 19 history of the 369 patent?
- 20 A Glanced over it, yes.
- Q Okay. And did you look at -- in preparing 21
- 22 for your deposition here today, did you review any
- 23 other exhibits that are not listed in your materials
- 24 considered for your declarations?
- 25 A Not that I recall.

- 1 Exhibit 3 for identification purposes. This is the
- 2 consultant CV that was attached to -- as an exhibit
- 3 to each of your declarations; is that right?
- A That's correct.
- 5 Q All right. Again, anything that you would
- 6 like to add to this CV?
- 7 A No. I mean, I've published additional
- 8 papers, but I -- I think this is fine.
- Q You've published additional papers since you
- 10 submitted this declaration in -- on December 9th or
- 11 December 10th of 2014?
- 12 A Yes. We publish all the time.
- 13 Q Okay. Apart from any other papers, any---
- 14 anything that you'd want to add to this --
- 15 A No.
- Q -- CV? 16
- 17 A No.
- 18 Q Anything that you'd want to delete from the
- 19 CV?
- 20 A Oh, no.
- 21 Q Any changes at all that you would like to
- 22 make?
- 23 A I mean, if there's a typo somewhere here, I
- 24 would like to change it, but I have not seen a typo.
- 25 If there is any misspelling or, you know, some

Page 15

- 1 Q Okay. Did you look at the patent owner's
- 2 response in the 053 case?
- 3 MR. PLUTA: Object to form.
- THE WITNESS: 053 case?
- 5 BY MR. TUMINARO:
- 6 O Yes.
- 7 A In preparation, I only looked at what's
- related to these two patents.
- Q Okay. Have you seen the patent owner's
- 10 response in the 053 case?
- 11 MR. PLUTA: Object to form, object to
- 12 relevance.
- 13 THE WITNESS: I don't understand your
- 14 question, Counsel. You mean since my deposition,
- 15 there was a response; is that what you're saying?
- 16 BY MR. TUMINARO:
- 17 Q Yeah. Since your deposition, there was a
- 18 response in the 053 case. Have you seen that
- 19 response?
- 20 A No.
- 21 MR. TUMINARO: Okay.
- 22 (Exhibit 3 was marked for identification
- 23 by the court reporter and is attached hereto.)
- 24 BY MR. TUMINARO:
- Q Sir, you've been handed what's been marked as

- 1 cosmetic things, you know, I might want to change
- 2 that. But I don't see anything that I can think of
- 3 to be changed here.
- Q Okay. Do you have any other CVs that you use
- 5 in your professional capacity?
- A Yes. 6
- 7 Q Apart from this consultant CV?
- 8 A That's right.
- 9 Q How many other CVs do you use?
- A Different venues; different CVs. I have a CV 10
- 11 for National Science Foundation grants. Those are
- 12 two-page CVs. I have a CV for merit and promotions
- 13 at the University of California. That's over
- 14 70 pages.
- Q Do you have any other CVs besides those two
- 16 that you mentioned?
- 17 A There are other ones for different
- 18 applications. You know, the -- the way they ask
- 19 you, a one-page CV or like an executive summary or
- 20 something like that, yeah.
- Q Do you -- do you talk on a regular basis,
- 22 give -- give presentations on a regular basis?
- A Yes, I do. 23
- 24 Q Do you have a CV that you use to indicate
- 25 your experience with talking engagements?

5 (Pages 14 - 17)

Page 18 Page 20 A No, because they know me so they don't ask THE WITNESS: We did. We did, yeah. 1 1 2 for a CV. 2 BY MR. TUMINARO: 3 Q Okay. 3 Q Did you actually do that work? A That's right. It would be a little weird for 4 A I'm sorry? 5 them to ask for a CV for somebody they're inviting 5 Q Were you the one that did that work? for a talk. 6 A I was the key engineer, yes. 7 Q Okay. 7 Q Okay. Is it -- you've worked on digital 8 A That's usually not done. 8 signal processors? 9 Q Is your NSF CV publicly available? 9 A Sure, yes. A That's a good question. I don't know if it 10 Q DSP for short; is that right? 11 is publicly available. 11 A Yes. Q Do you -- you're a -- you're a professor at 12 12 Q You've worked on system-on-a-chip 13 architectures? 13 the University of California, Irvine; is that right? 14 A I have, and I continue to, yes. 15 15 Q Do you publish your CV on -- on the Q You've continued to work on that? 16 university's webpage? 16 A Uh-huh. Q You've published papers on a system-on-a-chip 17 A No, we don't. We just have a summary on the 17 18 website. If you Google, you'll find out. 18 architecture? 19 Q So based on your CV, you've -- you've never 19 A All the time. 20 worked at a graphics processing company, correct? 20 Q You continue to do research on system-on-a-MR. PLUTA: Object to form. 21 21 chip architecture? 22 THE WITNESS: Well, I worked for our 22 A That's what we do. 23 23 start-up, and one of the application was for Q Okay. And you also work on processor 24 graphics processing. 24 architecture in general? 25 BY MR. TUMINARO: 25 A That's all my life, yes. Page 19 Page 21 Q The start-up being the MorphoSys? 1 Q That's your research? 1 2 A Yes, Morpho Technology. 2 A Yes. 3 Q Morpho Technologies. 3 Q You've written papers on processor 4 architecture? Okay. If we look at Morpho Technologies on 5 A Yes. 5 your CV, it says: "Duties: Cofounder; DSP design 6 Q Okay. All right. Let -- let's -- let's look 6 at your papers, if you would. On page 5 of your CV, 7 7 for communication and multimedia there's a section that reads "Journal" -- well, it 8 systems." 9 starts with "Publications." 9 Is that right? What about that signifies 10 Do you see that? 10 that you worked on graphics? 11 A So we added the functionality of multimedia A Yes. 12 graphics processing, because we were seeking 12 Q And in the "Journals" section, there are 13 customers and clients to adopt a technology. And we listed, starting on page 5 and spanning to page 13, 14 had a -- you know, we had a hammer. We were looking 14 88 journal articles; is that right? 15 A Right. 15 for a nail. Q And then starting on page 13, there's a 16 So basically, we had a parallel processor, an 16 section that reads "Journal" -- "Journals (other)." 17 SIMD, to say it more specifically. We were very 17 18 Do you see that? 18 proud of it because it worked really nicely. And 19 many of these problems are data parallel, so we felt 19 A Yes. 20 Q What does that refer to? 20 very strongly about being able to apply it to the 21 A These are not -- okay. Yeah. When -- when 21 pixel processing and so on. 22 you have a special issue in a journal -- actually, I 22 Q Okay. So you felt strongly that you could

6 (Pages 18 - 21)

24

see a typo here.

When you see a special issue in a journal,

25 then there would be editors that would be writing a

23 apply it to pixel processing. Did you actually

MR. PLUTA: Object to form.

24 apply it to pixel processing?

Page 22 Page 24 1 summary, an overview of what the special is all --1 graphics processing, right? MR. PLUTA: Object to form. 2 special issue is all about. So the "Journal 2 3 THE WITNESS: I don't know. 3 (others)" means that we wrote an opening statement 4 BY MR. TUMINARO: 4 about what this particular special issue is about. 5 Q Well, I counted, and there's only eight of 5 For instance, let's say there was a special issue for configurable computing, like jo3. them. We could go through them if you'd like. If So Fadi Kurdahi and I, plus others, wrote an 7 you look at j54. opening remark. So this is really not A Okay. I appreciate your due diligence on 9 this. I -- I would be glad to verify them. peer-reviewed. That's why it's called "Journals 10 (other)." Q On page 10, j54. 11 Q I see. 11 A Okay. Okay. j54. 12 Can I -- I just want to put a checkmark to --12 A All the other ones are peer-reviewed, and we 13 j54. Okay. 13 really don't count these for promotions. And this 14 is just a service to the society, IEEE society, and 14 Q Is that directed to a graphics paper? 15 A It is. 15 whoever else is in charge. We have to do that. I -- I don't know how it is in other fields, 16 Q Okay. How about j71? Is that a graphics 16 paper on page 11? 17 but it takes time because you have to review those 18 selected papers and make a summary of what -- what's 18 A Yes. 19 coming up, and so it takes time. 19 MR. PLUTA: Object to form. 20 BY MR. TUMINARO: 20 Q Okay. So under "Journals (other)," you have 21 six articles that are listed there; is that right? 21 O It is? 22 A Yes, it is. 22 A Right. 23 23 Q Okay. And after "Journals," starting on Q Okay. How about c47, which appears on 24 page 13 of your CV, there's book chapters; is that 24 page 18 of your declaration -- I mean of your CV. I 25 apologize. 25 right? Page 23 Page 25 1 A That's right. A Yes. 1 2 Q And there's five listed book -- book chapters 2 Q c76 on page 21 of your CV, is that related to graphics? 3 that you've contributed? 3 4 A Correct. 5 Q All right. c81, page 21, is that related to 5 Q Okay. Then on page 14, there's conference papers, I -- I assume; is that right? Is that what 6 graphics? 7 that means? A Yes. 8 A Yes. Q c87 on page 22, is that related to graphics? 9 Q They're refereed? 9 A Yes. 10 10 A That's right. Q c89, is that related to graphics? 11 Q Okay. And starting on page 14 and spanning 11 A Yes. 12 all the way until -- I guess it's page 29, there's 12 Q And c95 on page 23, is that related to 13 13 160 conference papers listed on your CV? graphics? 14 A Yes. 14 A Yes. 15 Q Okay. Then starting on page 29, there's 15 Q Those are eight papers that you say are 16 another section that reads "Technical Reports," related to graphics. 16 17 correct? 17 Are there any others that are listed on your 18 A Correct. 18 CV? 19 Q Okay. And from page 29 to page 30 of your 19 MR. PLUTA: Object to form. 20 CV, there's 13 technical reports that are listed? 20 THE WITNESS: Yes. 21 21 BY MR. TUMINARO: 22 22 O Where? Q So in total, if you add up all those papers 23 on your CV, you have 272 papers listed on your CV? 23 A So let's start. 24 Q If you'd --A Yes. 24 25 Q Okay. And only eight of them are directed to 25 A Yes.

7 (Pages 22 - 25)

Page 26

1 O -- take a look at it and tell me if there's 2 others.

A Sure. So some of these are tangentially

- related. Like j7, "Finding circular shapes in an
- 5 image on a pyramid architecture." Some of these
- techniques could be used to create a primitive, like
- primitives for 3D graphics because you're trying to
- figure out the shape. And primitives could be
- triangles or could be some other exotic shape. So
- 10 that's -- that -- those algorithms are useful, so I
- would consider that as one related to that. 11
- 12 So j12, although it says "image processing,"
- 13 but the concept of a hierarchical pyramid
- 14 architecture, it's very relevant to the graphics
- 15 because of the way -- as -- as you may recall, we do
- 16 the vertex processing, and then it goes down to the
- 17 pixel processing.
- 18 So this hierarchical level works for image
- 19 processing and graphics. And people have looked at
- 20 this hierarchical architecture. And to some extent
- a unified shader probably is trying to do something
- 22 like that. So I would put a little bit of
- 23 checkmark, at least from my point of view.
- 24 So let me finish all of this, and then if you
- 25 have any questions --

Page 27

- 1
- 2 A -- I will answer that. But I'd like to go
- 3 through this.
- O Okav. You want to --
- 5 A It would not take much time.
- 6 Okay. Sure. Go -- go right ahead. Please
- 7 do.
- A So the j14, also circle detection, it goes
- back to primitives and identifying primitive
- objects, although usually people use triangles, but
- that helps consolidate, you know, triangles or
- 12 whatever into circles. So that is -- that could be
- 13 used for graphics, as well.
- 14 Q Well, I'll stop you. In that paper, was it
- 15 actually directed to graphics?
- MR. PLUTA: Object to form. 16
- 17 THE WITNESS: It was for identifying circles.
- 18 BY MR. TUMINARO:
- Q Okay. And you mentioned that circles could
- 20 be used in graphics, but was it actually -- in that
- 21 paper, were you discussing graphics processing?
- 22 MR. PLUTA: Object to form.
- 23 THE WITNESS: Not directly. Indirectly.
- 24 BY MR. TUMINARO:
- 25 Q Okay.

1 A Again, j19 is about pyramid architectures.

- 2 So that's -- let me go through this more here.
- 3 So I'm down to j40, j41.
- 4 So I -- let me -- before forgetting, what we
- 5 did at Morpho, as you can imagine, it was a
- start-up. We had investors, and we had -- it was
- not possible to publish some of that work. So we
- did a lot of work on using SIMD for graphics
- processing, in particular, pixel processing, and
- they were not publishable.
- I just wanted to mention that to you for the
- 12 record, because there is no -- any documentation on
- 13 that. And that's true for many companies, by the
- 14
- 15 Q Okay. And before you go on, I notice that
- 16 some of these papers, you're listed as the first
- named author, and some you're -- you're the second
- or third or fourth or fifth named author.
- 19 A Uh-huh.
- 20 O How is that determined, what order that
- 21 you're an author?
- 22 A So that's a very interesting point of view
- 23 for faculty. All of these are our ideas or, in this
- case, my ideas. It's just when a student is working
- with me, we try to give him more credit. Some

Page 28

- 1 faculty don't follow this strategy. I do and many others do.
- 3 So we put their names first and ours in the
- 4 back, kind of in the middle or at the end. And if
- they help write some of this, they will get some
- credit for that, as well. We do writing, too, but
- it's the style of the faculty. I would say
- 75 percent follow what I do. Some like to see their
- 9 name first.
- 10 O So if you're listed as a first named
- inventor -- I mean, first named author, I apologize,
- that means you did more of the work on that paper?
- 13 MR. PLUTA: Object to form.
- THE WITNESS: Not really, no, no, because the 14
- 15 ideas come from us.
- 16 BY MR. TUMINARO:
- 17 Q Come from us, who's us?
- 18 A The faculty.
- 19 Q Okay.
- 20 A Me in this case. We give the ideas. We
- provide the -- I mean, the whole idea of MorphoSys
- architecture was my idea. But writing the code and
- implementing it, some of the students helped out.
- 24 Q Okay. All right. So then I'll ask, why
- 25 don't you go through your -- your papers, and if you

8 (Pages 26 - 29)

Page 30 Page 32 1 find any additional ones that you think relate to 1 A Yes. 2 graphics --2 Q Okay. So before you got involved in these 3 A Sure. I'm almost done. 3 cases between LG and ATI at the patent office, had Q -- you can let me know. you ever -- well, strike that. A So again, I repeat that what happened in 5 Before you got involved with these cases, LG Morpho was not published related to graphics. versus ATI, you never worked on a graphics case; is c78, it's "Hardware Accelerated Voxel 7 that right? Carving." Voxel is a three-dimensional pixel, so it 8 MR. PLUTA: Object to form. 9 THE WITNESS: No, that's not correct. is definitely graphics. And it's published in BY MR. TUMINARO: graphics -- computer graphics symposium, so Q That's not correct. 11 definitely that should be counted in your list, 11 12 among other things that I've mentioned, c78. 12 Which one of the cases listed on your CV 13 Q j78, you mean? 13 relates to graphics? 14 A c78. 14 A Fish and Richardson, RIM ITC versus GPH. 15 Q c78. 15 There's also one more, but it's just -- the A And you can see that it says "computer 16 Milberg -- 2011, Milberg, Class versus NVIDIA. 16 17 graphics symposium." 17 Q Any others? 18 "Recovering 3D Metric Structure and Motion 18 A I just looked at it quickly. And that's 19 for Multiple Uncalibrated Camera;" that's also 19 about it, I think, for now. Q Okay. When you were at Morpho Technologies, 20 computer graphics, c83. 20 21 I think you said eventually you added more graphics 21 There are a couple of papers here. We tried 22 to improve the SIMD to have branches. And that was capabilities to the MorphoSys? 23 in preparation for applying it for graphics 23 A It -- it had the capability. We just tried computation, because you want to do a conditional different applications on it to just be able to do statement in SIMD, which we did not have before. pixel processing and so on. You kind of -- you can Page 31 1 And that's what c85 with Anido and also with see that a graphics computation is really an array, 2 c81 which you already identified. to the array. And the MorphoSys was perfectly 3 "Persepolis: Recovering History with a 3 designed for that purpose. 4 Handheld Camera," that is definitely graphics you're Q Switching gears, I'd like to look at your 5 declaration, Exhibit 1. 5 trying to identify. 6 "Image Based Mesh Reconstruction and 6 A Okay. Rendering," that's also --7 7 Q Starting on page 11 of that declaration, Q You're saying c90? there is a large heading number III, "Technology 9 A c90, c91, "Camera Calibration Long Image 9 Background." 10 10 Sequences," those are all related to graphics. A Yes. 11 "Automatic creation of three-dimensional 11 Q Okay. In this technology background, it 12 avatars," that's also graphics. 12 spans all the way to page -- the bottom of page 16; 13 is that right? 13 Again, c98, anything with avatars, you're 14 trying to do -- carving these three-dimensional 14 A Yes. Q Oh, actually, there's one word at the top of 15 pixels, some of the details, I mean -- yes, c100, 15 16 c98. 16 page 17, "processing"; is that right? 17 A Yes.

17 So I would say there are at least twice as many as you mentioned that are directly or 18 19 indirectly related. 20 Anyway, so... 21 Q Okay. Let's look at your litigation support

22 experience that starts on page 2 of your --

23 A Yeah, sure.

24 Q -- CV, and that spans to page 5 of your CV; 25 is that right?

citation; is that right? MR. PLUTA: Object to form.

22 THE WITNESS: I can check.

23 I don't see one, yes.

24 BY MR. TUMINARO:

25 Q Where did the information for this section

Q Okay. In this entire section about the

technology background, there's not a single

9 (Pages 30 - 33)

18

20

21

Page 34 Page 36 1 come from? 1 THE WITNESS: I don't have a product. I just 2 A My knowledge of the field. 2 remember that. 3 Q Okay. So as an expert in graphics, you're 3 BY MR. TUMINARO: 4 knowledgeable about graphics technology? 4 Q You just remember that? A Yes. 5 A Right. Q You're knowledgeable about the evolution of 6 MR. TUMINARO: I'll like to have that marked. 7 graphics technology? 7 (Exhibit 4 was marked for identification A Yes. 8 by the court reporter and is attached hereto.) 9 Q If you turn with me to page 27, there is a 9 BY MR. TUMINARO: 10 Figure 1 that is shown in --Q I've handed you what's been marked as 11 A Page 27? 11 Exhibit 4 for identification purposes. This is a 12 Q I'm sorry, I meant paragraph 27. If I said 12 document entitled "How GPUs Work." 13 page 27 --13 Do you see that? 14 A Right. 14 A Yeah. 15 15 Q Paragraph 27, there's a figure shown in Q And if you'll turn with me to page -- the 16 second page of -- of Exhibit --16 paragraph 27. 17 A Correct. 17 A Uh-huh. Q -- 4. 18 Q Where did that figure come from? 18 19 A From one of my sources that I had or what's 19 MR. PLUTA: I'm just going to object to the 20 on the web. introduction of this evidence as irrelevant at this 21 Q Did you generate this figure? point, lack of foundation. 22 A It's from one of my references or from what's 22 MR. TUMINARO: I'm trying to get to the 23 foundation. 23 available on the Internet. It could be from a specific source. I -- I -- I did not exactly draw 24 Q If you look with me to the right of page --25 this. 25 it's listed 127, in the right-hand column there's Page 35 1 Q Okay. So this came from something on the a -- sort of midway through there's a sentence that 2 Internet or some other source? starts with "for example." A Yeah. 3 Do you see that? Q If you look at your materials considered, 4 A Yes. 5 O I'll read it for the record: 5 there's -- that Internet or other source is not 6 listed, is it? 6 "For example, the NVIDIA GeForce A That source? No. 7 3, launched in February 2001, Q Okay. If you turn with me to paragraph 31 of 8 introduced programmable vertex 9 9 your declaration -shaders. These shaders provide 10 A Yes. 10 units that the programmer can use 11 11 for performing matrix-vector Q -- the first sentence of paragraph 31 reads: 12 "The first programable pipeline 12 multiplication, exponentiation, and 13 13 (PP) was introduced in 2001." square root calculations, as well as 14 14 Do you see that? short default program that use these 15 15 A Yes. units to perform vertex Q What's the basis for that statement? 16 transformation and lighting." 16 A Just my knowledge, what I -- what I know from 17 17 Do you see that? 18 what I've read. 18 A Yes. 19 Q What pipeline was it that you were referring 19 Q Is that the product that you were referring 20 to there? 20 to that was a programmable pipeline in 2001? 21 A I'm not sure. I'm not sure if that was the A Used for graphics. 22 Q Well, what was the product? Is there a 22 product or there were other products, but it seems 23 product that's associated with that? to be consistent with what I said. 24 24 A No. Q Okay. So when you wrote this sentence in paragraph 31, you didn't have any specific product 25 MR. PLUTA: Object to form.

10 (Pages 34 - 37)

	Page 20		D 40
1	Page 38 in mind?	1	Page 40 A The
2	A No, no.	2	MR. PLUTA: Object to form.
3	Q Just your general knowledge of the GPU	3	THE WITNESS: The first
4	evolution in technology?	4	MR. PLUTA: Lack of foundation.
5	A Correct.	5	THE WITNESS: I could not tell you, but
6	Q If you turn with me to paragraph 32 of your	6	the DirectX 10 does because it's the latest one or
7	declaration.	7	one of the latest ones that we have. So so I
8	A Yeah.	8	would say I can't tell you if it was the first
9	Q The first sentence reads:	9	one or not. I cannot tell you that.
10	"OpenGL and DirectX are now the	10	MR. TUMINARO: I'll have this marked.
11	common standards that most hardware	11	(Exhibit 5 was marked for identification
12	vendors support as part of their	12	by the court reporter and is attached hereto.)
13	graphics card development	13	BY MR. TUMINARO:
14	environment."	14	Q So you've been handed what's been marked as
15	Do you see that?	15	Exhibit 5. This is titled "Exploiting the Shader
16	A Yes.	16	Model 4.0 Architecture."
17	Q When you say "now," what time frame are you	17	Do you see that?
1	referring to?	18	A Yeah, yes.
19	A Well, now meant December 2014.	19	Q And it says in the abstract I'll read
20	Q Okay. So you're aware of your general	20	the first sentence says:
21	knowledge of GPUs that at the time that the 871	21	"The Direct 3D10/SM4.0 system is
22	*	22	the 4th generation programmable
1	provided a unified shader architecture, correct?	23	graphics processing units (GPUs)
24	MR. PLUTA: Object to form.	24	architecture. The new pipeline
25	THE WITNESS: So am I aware that at the time	25	introduces significant additions and
	Page 39		Page 41
1	frame that 871 was filed, there was a unified shader	1	changes to prior generation
2	frame that 871 was filed, there was a unified shader or not?	2	changes to prior generation pipeline."
2 3	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO:	2 3	changes to prior generation pipeline." And then skipping a sentence, it says:
2 3 4	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There	2 3 4	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced
2 3 4 5	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry.	2 3 4 5	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified
2 3 4 5 6	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your	2 3 4 5 6	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common
2 3 4 5 6 7	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the	2 3 4 5 6 7	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable
2 3 4 5 6 7 8	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed	2 3 4 5 6 7 8	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages."
2 3 4 5 6 7 8 9	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is	2 3 4 5 6 7 8 9	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on.
2 3 4 5 6 7 8 9 10	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL	2 3 4 5 6 7 8	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that?
2 3 4 5 6 7 8 9 10	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct?	2 3 4 5 6 7 8 9	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay.
2 3 4 5 6 7 8 9 10	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry.	2 3 4 5 6 7 8 9 10 11	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that?
2 3 4 5 6 7 8 9 10 11 12	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct?	2 3 4 5 6 7 8 9 10 11 12	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the
2 3 4 5 6 7 8 9 10 11 12 13	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10	2 3 4 5 6 7 8 9 10 11 12 13	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of
2 3 4 5 6 7 8 9 10 11 12 13 14	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified	2 3 4 5 6 7 8 9 10 11 12 13	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay.
2 3 4 5 6 7 8 9 10 11 12 13 14 15	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right?	2 3 4 5 6 7 8 9 10 11 12 13 14 15	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form.	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report?	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report? BY MR. TUMINARO:	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to that?
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report? BY MR. TUMINARO: Q I'm asking you the question.	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to that? BY MR. TUMINARO:
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report? BY MR. TUMINARO: Q I'm asking you the question. A It's not in here, so I don't I did not opine on that, but, you know Q So based on your general knowledge of	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to that? BY MR. TUMINARO: Q I was going to ask the question. A Right. Q So does this does this jog your memory, as
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report? BY MR. TUMINARO: Q I'm asking you the question. A It's not in here, so I don't I did not opine on that, but, you know Q So based on your general knowledge of graphics, you're not aware one way or the other	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to that? BY MR. TUMINARO: Q I was going to ask the question. A Right. Q So does this does this jog your memory, as an expert in graphics, that Shader Model 4.0 in
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report? BY MR. TUMINARO: Q I'm asking you the question. A It's not in here, so I don't I did not opine on that, but, you know Q So based on your general knowledge of graphics, you're not aware one way or the other whether DirectX 10 was the first API to have a	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to that? BY MR. TUMINARO: Q I was going to ask the question. A Right. Q So does this does this jog your memory, as an expert in graphics, that Shader Model 4.0 in DirectX 10 was the first to introduce a unified
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	frame that 871 was filed, there was a unified shader or not? BY MR. TUMINARO: Q There A Can you repeat the question? I'm sorry. Q Sure. So you were aware, based on your general knowledge of GPUs that at the time that the 871 patent was filed A Which is Q which is 2003, neither DirectX nor OpenGL provided a unified shader architecture, correct? A I can't be certain on that one, sorry. Q How about this? You're aware that DirectX 10 was the first version of DirectX that had a unified shader model, right? MR. PLUTA: Object to form. THE WITNESS: Did I say that in my report? BY MR. TUMINARO: Q I'm asking you the question. A It's not in here, so I don't I did not opine on that, but, you know Q So based on your general knowledge of graphics, you're not aware one way or the other	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	changes to prior generation pipeline." And then skipping a sentence, it says: "The main facilities introduced that we ponder upon are, Unified Architecture providing common features set for all programmable stages." And then it goes on. Do you see that? A Okay. MR. PLUTA: I'm going to object to the relevance of this exhibit and also lack of foundation. Also, hearsay. THE WITNESS: I'm seeing this for the first time, so I mean, judging from that paragraph, it says what it says. What do you want me to add to that? BY MR. TUMINARO: Q I was going to ask the question. A Right. Q So does this does this jog your memory, as an expert in graphics, that Shader Model 4.0 in

11 (Pages 38 - 41)

Page 42 Page 44

- 1 MR. PLUTA: Object to form.
- 2 THE WITNESS: I could not answer that based
- 3 on this document. I have to do more research on
- 4 that.
- 5 BY MR. TUMINARO:
- Q So you don't know one way or the other; is
- 7 that your testimony?
- A That is not my testimony. I need more work
- on that. I have some ideas, but I'm not going to
- render my ideas based on just one article.
- Q Well, what's your idea, then --11
- 12 (Simultaneous speaking - unreportable.)
- 13 MR. PLUTA: Object to form, calls for
- 14 speculation.
- 15 THE WITNESS: I need more work on that to
- 16 give you an answer.
- 17 BY MR. TUMINARO:
- 18 Q You can't -- do you have an idea of when the
- 19 first unified shader architecture came out?
- 20 A I can't give you for certain what date.
- 21 Q What's your best understanding of what the
- 22 date is?

1

- 23 A I can't guess.
- 24 Q I -- I didn't ask you to guess. I asked,

MR. PLUTA: Object to form.

25 what's your best understanding?

- 1 felt. We just looked at their architecture; we
- 2 still do. We look at their papers. I'm interested
- in architectures. I'm not interested in how they
- 4 view and what they interpret what is possible or
- 5 not.
- 6 BY MR. TUMINARO:
- 7 Q And just to be clear, you didn't consider
- that in forming your declar- -- the opinions
- expressed in your declaration; is that right?
- MR. PLUTA: Object to form.
- THE WITNESS: I did not mention it here. 11
- 12 BY MR. TUMINARO:
- Q So given that now the two major APIs, OpenGL
- 14 and DirectX 10, provide a unified shader model, it
- would be more efficient for graphics hardware to
- implement a unified shader; isn't that right?
- 17 MR. PLUTA: Object to form.
- 18 THE WITNESS: What do you define efficiency?
- 19 BY MR. TUMINARO:
- 20 Q How would you define efficiency?
- A You asked me the question about efficiency.
- What efficiency do you have in mind?
- Q Well, that's what I'm asking you, how do you
- understand the word "efficiency" in the context of
- graphics processing?

Page 43

- 2 THE WITNESS: I'd have to guess, and I'm not
- 3 going to guess.
- 4 BY MR. TUMINARO:
- Q Okay. You are aware, though, now, as of
- 6 today, OpenGL and DirectX provide a unified shader
- architecture?
- 8 MR. PLUTA: Object to form, relevance.
- THE WITNESS: Yes.
- 10 BY MR. TUMINARO:
- 11 Q Okay. And you know that -- you've heard of
- 12 NVIDIA?
- Q And you understand that NVIDIA initially was 14
- 15 skeptical of whether unified architecture would
- 16 work?
- 17 MR. PLUTA: Object to form, relevance.
- THE WITNESS: I have no idea about their 18
- 19 position on that.
- 20 BY MR. TUMINARO:
- 21 Q So based on your general knowledge of shader
- 22 architecture and graphics evolution, you have no --
- 23 no understanding one way or the other?
- 24 MR. PLUTA: Object to form.
- 25 THE WITNESS: I would not guess what NVIDIA

- Page 45 A This is a very broad question. What is --
- 2 what is the application?
- 3 Q Graphics processing.
- A Counsel, there's a graphic processing in your
- 5 smartphone. There's a graphic processing in your
- 6 desktop.
- 7 Q Okay.
- 8 A There's a graphic processing for applications
- in military and civilian applications. Which one do
- you have in mind? You're asking a very broad
- 11 question.
- 12 Q Okay. So I have in mind graphics processing
- 13 in my cell phone or in a desktop.
- A You are asking that?
- 15 Q Yeah, in either one of those, what would it
- 16 mean to be efficient?
- 17 A Okay. Efficiency means computation
- 18 efficiency. Efficiency could be battery power
- efficiency. And between the two applications, I
- would think that you agree with me that the
- smartphone battery power efficiency's a lot more
- 22 important than your computation efficiency.
- Efficiency could mean cost. It's more
- efficient to have -- well, maybe you don't use the
- word "efficiency" for cost, but it means lower cost

12 (Pages 42 - 45)

Page 48

- 1 that gives you the targeted efficiency. So it
- 2 really depends.
- 3 If your -- if your unified shader is going to
- 4 cost you more for grandma's phone, I might opt for
- 5 a -- just a hardwire approach. Just it's cheaper.
- 6 I might just do that. I don't think NVIDIA or
- 7 anybody will -- will dispute that.
- 8 So it really depends on what your target
- 9 application is and what your cost is. If the cost
- 10 of unified -- unified shader comes down to the level
- 11 of that is acceptable to many target applications,
- 12 then that would be the way to go.
- 13 Q You mentioned that if you were going to put
- 14 it in grandma's phone, you might want to just use a
- 15 hardwire approach.
- 16 A Uh-huh.
- 17 Q What's a hardwire approach?
- 18 A It's the FFP, the fixed function pipeline
- 19 basically, or used to be.
- 20 Let's say the unified shader is very exotic.
- 21 It probably is all the latest technologies that you
- 22 have, but I really don't want that. I really don't
- 23 want -- you don't want to have an 8-cylinder car.
- 24 You're just going to the grocery store, and that's
- 25 all you're doing. You're not racing. That would be
 - Page 47
- 1 the grandma's. Then you may want to use -- then you
- 2 may want to use an ASIC approach, basically,
- 3 where -- where you just -- simple and already
- 4 proven, and it's very efficient cost-wise and
- 5 probably power-wise, too.
- 6 So it really depends. And I don't think
- 7 anybody would dispute that.
- 8 Q So are you suggesting that graphics
- 9 processors are now moving toward more of a hardwire
- 10 approach?
- 11 MR. PLUTA: Object to form.
- 12 THE WITNESS: That's not what I said. I said
- 13 it depends on your application. A unified shader
- 14 gives you the capability through software means.
- 15 It's a lot more advanced. It's the way to go. But
- 16 there are applications that you may find out that
- 17 unified shader is too much of a burden in terms --
- 18 in terms of power consumption, cost and so on.
- 19 If the cost and power consumption comes down
- 20 towards the ASIC solutions, then yes, that would be
- 21 the ultimate graphics engine.
- 22 BY MR. TUMINARO:
- 23 O So you're --
- 24 A We're struggling with this right now in
- 25 all -- all the things that we do that are

- 1 programmable.
- 2 Q So you're aware that in graphics processing,
- 3 initially it was fixed pipeline, correct?
- 4 A Yes.

Page 46

- 5 Q Okay. And then it moved to a programmable
- 6 pipeline, correct?
- 7 A That's right.
- 8 Q And unified pipeline is now sort of the --
- 9 A Program.
- 10 Q -- state of the art kind of pipeline?
- 11 MR. PLUTA: Object to form.
 - THE WITNESS: Yeah.
- 13 BY MR. TUMINARO:
 - Q And you said it's the way to go; is that
- 15 right?

12

14

- 16 A How often do you charge your phone? You
- 7 charge it a lot. You've got an 8-core processor on
- 18 your phone, so we are getting to a point of wanting
- 19 to know the efficiency of the execution. A
- 20 programmable solution is good for lots of people.
- 21 I'm not saying we're going back to the ASIC
- 22 solutions. I'm just saying when you talk about
- 23 efficiency, you have to be more accurate in terms of
- 24 what is -- what problem are you trying to solve?
- 4 what is -- what problem are you trying to solve? 5 You have to -- you have to qualify that. What --
 - Page 49
- 1 what do you mean by efficiency. That's -- that's
- 2 what I'm trying to tell you.
- 3 I'm not saying we're dropping off unified
- 4 shader or programmable solutions. I'm not saying
- 5 that at all. If I said that, I maybe mis---
- 6 misspoke.
- 7 Q So you're aware in your understanding of
- 8 graphics processing that the industry has moved away
- 9 from a fixed pipeline, right?
- 10 MR. PLUTA: Object to form, lack of
- 11 foundation.
- 12 THE WITNESS: I said as much in my report,
- 13 but I'm saying if -- if efficiency -- power
- 14 efficiency's important, there might be solutions
- 15 that are not that way. That's what I'm trying to
- 16 tell you.
- 17 BY MR. TUMINARO:
- 18 Q Okay. My question was not about efficiency.
- 19 I'm just saying, are you aware that it moved -- the
- 20 industry has moved away from fixed pipeline and now
- 21 it's toward a programmable pipeline?
- 22 MR. PLUTA: Object to form --
- 23 BY MR. TUMINARO:
- 24 Q You said that in your declaration, correct?
- 25 A Correct. But your original question was

13 (Pages 46 - 49)

Page 50 Page 52 1 about efficiency, if you recall. Are we -- are we 1 programmable unified shader is the way to go because 2 out of that question? 2 people don't want to design new chips every time Q Just to remind you how this works, I ask a 3 there's a bug. So I hope you understand my position question; you answer the question that I ask; is 4 on that. that right? 5 BY MR. TUMINARO: A Yes. Then I'm confused about what question Q Okay. Switching topics, I'd like to talk 7 you're asking. about -- you've heard of what a -- a register? You Q Okay. So now I'm asking the question. know what a register is? 9 You're aware that the industry has moved away A Register in the context of? 10 from a fixed pipeline and has moved toward a 10 Q Computer processing. 11 programmable pipeline, correct? 11 A Yes. MR. PLUTA: Object to form. 12 12 Q What's a register? 13 THE WITNESS: I said as much in my report. 13 A Stores information. 14 BY MR. TUMINARO: 14 Q Okay. A register is typically a multi-ported Q Okay. And the industry -- after the -- the 15 15 storage unit? 16 programmable pipeline is moving toward a unified A Typically. What's your application? 16 17 shader pipeline, is that correct, the graphics 17 Q Graphics processing. 18 processing industry? 18 A I mean, yes and no, depending on what you're 19 MR. PLUTA: Object to form. trying to do. I have to see what the diagram 20 THE WITNESS: That is correct. 20 looks -- schematic looks like to tell you what 21 BY MR. TUMINARO: 21 you're trying to do. 22 Q Okay. And the unified shader architecture, Q Okay. A register is typically the closest 23 it's the thing of the future, correct? piece of memory to an ALU? 24 A What do you --24 MR. PLUTA: Object to form. 25 MR. PLUTA: Object to form. 25 THE WITNESS: Generally, yes. Page 51 Page 53 THE WITNESS: What do you mean by the 1 BY MR. TUMINARO: 1 Q And a memory is typically something different 2 thing --3 BY MR. TUMINARO: 3 than a register, correct? MR. PLUTA: Object to form. Q Well, you said it's the way to go. 5 THE WITNESS: I don't understand why you A If I talk about efficiency -- if your 5 6 power -- if your power consumption is an issue for 6 distinguish between the two. BY MR. TUMINARO: 7 7 you, we may find changes on that model. Q Well, a memory is typically further from an 8 BY MR. TUMINARO: 8 9 ALU compared to a register? Q Okay. How about this. What if your factor 10 MR. PLUTA: Object to form. 10 that you're considering is performance, not -- not 11 THE WITNESS: Proximity to ALU doesn't 11 power consumption, performance? If you want to 12 increase performance, would you move toward a 12 identify the differences. It just stores 13 unified shader architecture? information. Remember what I defined register to

14 be, stores information; so does memory.

15 BY MR. TUMINARO:

Q And a memory is typically a single-ported storage unit as compared to a register, which is

18 typically a multi-ported?

19 A No. You can have dual port memories, too. I

have designed dual port memories.

21 Q Well, let me ask you this. A register is a

22 different thing than a memory, right?

23 A In terms of storing information, no.

24 Q But in terms of difference -- they're

25 different, correct? A register is different than a

14 (Pages 50 - 53)

14

15

17

23

MR. PLUTA: Object to form.

16 this, but an ASIC solution is far more efficient in

18 it's not programmable because it will have problems 19 in case there's a bug or there are fixes to be made.

20 So that's why unified shader is better, because you

21 can change it, you can modify it easy, and you don't

But it is a well-known concept that the

24 hardwired solutions are more power efficient and

25 they're more computation efficient. But a

22 have to retape the device.

THE WITNESS: Okay. You may be shocked by

computation than a programmable unified shader, but

Page 54 Page 56

- 1 memory?
- A In what way? I don't -- I mean, they have
- 3 different names, yes, they are different names. But
- 4 they store information and you retrieve information.
- 5 So from that point of view --
- Q So then why does the industry use a
- 7 different --
- 8 MR. PLUTA: Hold on, Counsel. I don't --
- 9 MR. TUMINARO: I thought he was done.
- 10 MR. PLUTA: I don't think he was done.
- THE WITNESS: I wasn't done, yeah. Sorry. 11
- Let me identify what a block of storage is. 12
- 13 You write to it and you read from it. Both of these
- 14 entities do that. Register and memory are
- 15 indistinguishable in terms of writing to it and
- 16 reading it.
- 17 In terms of volatility, they both could be
- 18 nonvolatile or could be volatile, both of them. So
- I don't see any difference in terms of
- 20 functionality.
- 21 BY MR. TUMINARO:
- 22 Q Well, the industry has come up with two
- 23 different names for these storage units, right? A

Q So are you telling me that it's -- it's 2 typically not the case that a register is

A I don't know if I said that.

Q So is a -- is a register typically

A It depends on your application. I mean, you

Q So in your mind, is a register the same thing

THE WITNESS: The functionality is very

information for a period of time. From that point

A As in -- you asked the question earlier, it's

25 closer to the ALU or the processor, so that's the

8 could have an accumulator, which is a register, and

9 it could be one-ported. It really depends on what

10 application you have. It's really -- you cannot generalize that, application dependent.

MR. PLUTA: Object to form.

16 similar in terms of reading, writing, storing

18 of view, the same. So it stores data.

Q Are there any differences?

22 memory -- actually, a lot faster, yeah.

A So register is a little faster than a

- 24 register and a memory, correct?
- 25 A Correct.

3 multi-ported?

6 multi-ported?

as a memory?

19 BY MR. TUMINARO:

Q Anything else?

1

5

12

13

14

15

20

21

23 24

- 1 case, as well. So those are very important issues.
- Q Those are two differences. Any others
- 3 between a register and a memory?
- A The actual circuit design could be different
- 5 because you're trying to save power, save energy and
- so on. So yeah, if you are asking memory, it would
- 7 be a little different, but the functionality's
- 8 identical.
- 9 Q Okay. Anything -- any other differences that
- 10 you can think of between an AL- -- between a
- 11 register and a memory?
- A It stores data -- no, I think I mentioned all
- 13 the big differences. Performance, proximity to ALU
- and the circuit actually is being different.
- 15 Q Okay. You've heard of the term "ALU"?
- 16 A Yes, I have for a long time, unfortunately.
- 17 O ALU means what?
- 18 A Arithmetic logic unit.
 - Q Okay. You've heard of the term "processor
- 20 unit" -- strike that, "processor"?
- 21
- 22 Q Okay. Is a processor the same thing as an
- 23 ALU?

19

- 24 MR. PLUTA: Object to form.
- 25 THE WITNESS: Okay. Traditionally, if this

Page 55

- 1 is a class, which I will have in two weeks, a
 - 2 processor includes an ALU and register file and any
 - 3 other things that you can think of, a processor, a
 - 4 processor, a microprocessor.
 - 5 BY MR. TUMINARO:
 - Q So -- so you're saying a processor has
 - additional structure that's not in an ALU?
 - A If you define the microprocessor the way we
 - 9 define it in our textbooks, a microprocessor has
 - 10 those components.
 - 11 Q An ALU and register?
 - 12 A Register file and other things, address
 - 13 decoder and so on --
 - Q So --14
 - 15 A -- and program counter and so on. Sorry.
 - Q So an ALU can do functions like add, 16
 - 17 subtract, compare, correct?
 - 18 A Sure, yes.
 - 19 Q Whereas a processor can run instructions,
 - 20 right?
 - 21 A No. Remember when I put the -- the -- the
 - 22 area around the processor, the processor included
 - the ALU and the register file.
 - 24 Q So, what, you're saying a processor doesn't
 - 25 run instructions?

15 (Pages 54 - 57)

Page 58 Page 60 1 A I didn't say that. 1 will say the engine is different from the car. The 2 MR. PLUTA: Object to form. 2 car includes the engine. This is the same 3 BY MR. TUMINARO: 3 relationship. So it's the only way I can answer your question without confusing myself. Q So -- so my question was, a processor can run instructions, right? And you said no. So if the processor is the car, engine is the ALU, is that -- does that help you with -- to A So let me --7 7 Q So let me ask the question again. answer? 8 A Yeah, please. 8 Q That helps. But -- so just in your example, 9 9 a car is not the same thing as an engine, right? Q A processor can run instructions, right? 10 A A processor executes instructions from its 10 A It includes the engine. Q It includes the engine, but they're not the 11 instruction memory. And based on those 11 12 instructions, it will activate the control signals 12 same thing? 13 for all the internal components, which means A Where do you say the car is? I mean, I 14 activates the register file, activates the ALU and a 14 don't -- I don't understand. Would you call the 15 few other blocks. shell of the car to be the car? I don't know. Q Just so I'm clear because I'm not -- that 16 Q Okay. At least I understand your testimony. 16 17 means yes, a processor can run instructions; is that 17 A Okay. Thank you. 18 right? 18 MR. PLUTA: We've been going for about an 19 A I think I just said that, yeah. 19 hour. Is it a good time for a break? 20 20 Q I just wasn't clear. I didn't know if that MR. TUMINARO: Oh, yeah, sure, yeah, yeah. 21 THE VIDEOGRAPHER: We are off the record. 21 was a yes or not. 22 22 A The answer is the processor executes The time is 10:45 a.m. 23 instructions based on a program counter, which is 23 (Recess.) 24 24 basically sequencing through the instruction memory, THE VIDEOGRAPHER: We're back on the record. 25 and activates -- I'm giving you a little bit more 25 The time is 10:59 a.m. Page 59 Page 61 1 Please continue. 1 answer because --2 (Exhibit 6 was marked for identification Q Okay. 3 A -- I think it has to be clear that ALU does by the court reporter and is attached hereto.) 3 4 BY MR. TUMINARO: 4 the function on behest of the instruction decoded. 5 Q Welcome back, Dr. Bagherzadeh. Q Okay. So just to be clear, though, an ALU by 6 itself does not run instructions, right, or cannot 6 A Thank you. 7 Q You have in front of you Exhibit 6, which is 7 run instructions by itself? 8 MR. PLUTA: Object to form. 8 the Rich patent, U.S. patent number 5,808,690. 9 You recognize this document, right? THE WITNESS: Instruction has to be decoded 10 A Yes, I do. 10 to activate the control signals for the ALU. Some 11 people mistakenly call ALU the microprocessor. 11 Q You considered this, the Rich patent, in 12 That's -- that's their definition. But, yes, ALU 12 preparing your declarations --13 13 gets its control signals from the decoded A Yes. 14 Q -- in this case? 14 instructions. 15 (Exhibit 7 was marked for identification 15 BY MR. TUMINARO: by the court reporter and is attached hereto.) 16 Q Okay. So you would say it's a mistake to 16 17 call an ALU a processor? 17 BY MR. TUMINARO: 18 Q Okay. You also have in front of you 18 MR. PLUTA: Object to form. 19 THE WITNESS: The ALU by itself, you may call Exhibit 7, which is U.S. patent number 6,897,871. 20 You -- I'll refer to this as the 871 patent. 20 it the processor, but that's not what textbooks 21 You'll understand what I'm talking about? 21 usually talk about. 22 22 BY MR. TUMINARO: A Yes. 23 Q And you considered the 871 patent in 23 Q So the textbook definition would say that a 24 24 microprocessor is not the same thing as an ALU? preparing your declaration, correct?

16 (Pages 58 - 61)

25

A Yes.

A This is like saying that mechanical textbooks

Page 62 Page 64 Q Okay. And, in fact, just to be clear, it's Q Well, I -- I read your report, and that's 1 2 your opinion that Rich renders claim 15 of the 871 2 what I'm trying to understand is -- well, let me ask 3 patent obvious, correct? 3 you this. 4 Are you saying that general purpose register A Yes. Right. Rich teaches claim 15, correct. 5 Q Okay. Now, if we look at claim 15 of the 871 5 block claim -- in claim 15 of the 871 patent patent, there are sort of three elements recited. corresponds to memory 34 in Figure 2 of Rich? There's a unified shader comprising a general 7 A I was just about to answer that question, if purpose register block, a processor unit and a 8 you allow me. It will take a few minutes. 9 sequencer, right? 9 It does provide the resources to the ALU, and 10 A Yes. 10 you can see it from Figure 12. And that's consistent with providing operands to the ALU from 11 Q Okay. Now, I think I have this right. 11 You're saying that -- well, okay. Strike that. what -- what I --12 12 13 Let's look at Figure 2 of Rich. 13 THE REPORTER: "Providing" what? 14 A Uh-huh, yes. 14 THE WITNESS: Operands, o-p-e-r-a-n-d. 15 15 Q Figure 2 of Rich, it shows an architecture So that combination of memory in ALU that's disclosed, a computer architecture that's 16 satisfies that operands are available, and ALU 16 17 executes them. 17 disclosed in Rich, right? 18 A Correct. 18 BY MR. TUMINARO: 19 Q And there is an ALU 33 and a memory 34 in 19 Q Okay. And then --20 20 A And I will continue to --Figure 2 of Rich? 21 21 A Correct. Q Oh, there's more? Okay. 22 Q And there is a processing element array 22 A And Rich says: 23 23 control 40 in Figure 2 of Rich, correct? "Each processing element 32 24 A Correct. 24 comprises an 8-bit multifunction 25 Q Now, correct --25 arithmetic logic unit 33, directly Page 63 Page 65 1 A Processing element array controller, yes. 1 coupled to its own bank of" --2 Q Okay. Now, correct me if I'm wrong, but I --Too fast? 3 THE REPORTER: Yes. 3 I think this is what you're saying, that in your opinion, memory 34 corresponds to the claimed 4 THE WITNESS: Sorry. 5 general purpose register block; is that right? I repeat: A So let's look at -- let's look at the details 6 "Each processing element 32 7 of this. 7 comprises an 8-bit multifunction Q I'll help you out. You talk about Rich in 8 arithmetic logic unit 33, directly 9 9 claim 15 starting at paragraph 214 of your coupled to its own bank of 128 bytes 10 10 declaration. of memory, 34. Each ALU is capable How -- how about we do this? How about we do 11 11 of simultaneously accessing its own 12 this? 12 memory and can share data with its 13 neighbors via an interconnecting bus 13 A Okay. Yes. 14 14 Q How about what you can do for me is circle in architect structure." 15 Figure 2 of Rich what you say corresponds to the So -- so it does provide that functionality claimed elements in claim 15 of the 871 patent. 16 of a -- of a register in terms of providing 16 17 MR. PLUTA: Object to form. 17 operands. 18 THE WITNESS: I would rather not do that and 18 BY MR. TUMINARO: 19 answer your question, because this is a complicated, Q So when you say "it does," you're saying very high-level block diagram. And I -- I would not memory 34, in your opinion, corresponds to the 20 do service to the inventors and what they meant. general purpose of register block claimed in claim 21 22 22 But let me try to do it differently and just 15 of the 871? 23 MR. PLUTA: Object to form. 23 refer to you -- to -- refer you to my report and see 24 THE WITNESS: It provides the data that it 24 if that's satisfactory to you. 25 BY MR. TUMINARO: 25 needs in combination with other components.

17 (Pages 62 - 65)

Page 66 Page 68 1 BY MR. TUMINARO: 1 A Right. 2 Q You'll agree with me that Rich never Q Okay. What in Rich are you saying 3 corresponds to the claimed processor unit in claim discloses that vertex data is retrieved from memory 4 15 of the 871 patent? 4 34, right? 5 MR. PLUTA: Object to form. 5 MR. PLUTA: Object to form. THE WITNESS: So the processing units are THE WITNESS: It -- it is -- well, I mean, it 6 7 those -- those ALUs. That's what Rich says. 7 does perform the operations for graphics BY MR. TUMINARO: computation. And even if it doesn't say it 9 Q Okay. specifically, it is capable of doing that. 10 A The processing elements 32 operating --10 BY MR. TUMINARO: processing elements 32. It says that here --11 11 Q Okay. Let's --O Okav. 12 A Because -- because it's -- it is a processing 12 13 A -- "processing elements 32 13 element, it has access to the memory. And it is 14 operating normally as a single 14 done for interpolation of the pixels, rasterizations 15 15 and other functions. So, therefore, vertex instruction multiple data configuration." 16 16 processing is part of that, and therefore, it is 17 Q Okay. consistent with my understanding. 17 18 A Each processing element 32. So that -- that 18 Q Okay. Let -- let's look at what you said in 19 box is called a processing element, which includes 19 paragraph 216. The last sentence says: 20 an ALU. 20 "Rich does not explicitly disclose 21 Q Okay. Great. That's it? 21 vertex processing on data retrieved 22 A Yes. 22 from processing memory 34." 23 23 Q Okay. Now, what in Rich are you saying Do you see that? 24 corresponds to the claimed sequencer in claim 15 of 24 A Yes. 25 the 871 patent? 25 Q Do you agree with that statement? Page 67 Page 69 1 A The control unit 40: 1 A I say it. 2 "The processing element, array 2 Q So it's a true statement? 3 3 element control unit 40 is primarily A It's what I've said, yeah. 4 responsible for sequencing 4 O Okav. A But -- but it doesn't do it explicitly, but 5 5 instructions and addresses to the 6 it's implicitly. 6 processing element array 30." 7 7 Q So then you would agree with me that Rich Q Great. 8 does not disclose that vertex data is retrieved from A It's in my chart. 9 Q Excellent. We're on the same page now. 9 memory 34, right? 10 10 Thank you. MR. PLUTA: Object to form. A You are so excited, Counsel. Okay. Okay. 11 THE WITNESS: I think I can read what I have 11 12 Good. 12 here. It doesn't say it explicitly --Q All right. Now, you would agree with me that 13 13 MR. TUMINARO: Okay. 14 Rich never discloses that vertex data is retrieved 14 THE WITNESS: -- but for an engine that is 15 doing graphics computation, it is -- to me, that's 15 from memory 34, correct? MR. PLUTA: Object to form. 16 clear. 16 17 THE WITNESS: Well, this is a graphics 17 BY MR. TUMINARO: 18 processing engine, and clearly talks about Q Okay. But it's your opinion that it would be performing those operations, so it would have been obvious to modify Rich to store both vertex data and 20 very clear to whoever is looking at this technology 20 pixel data in memory 34; is that right? 21 that that's what Rich is talking about, that it can 21 A That is correct. 22 22 access vertex or pixel. Q Okay. But doesn't Rich actually teach away 23 from storing both vertex and pixel data in memory 23 BY MR. TUMINARO: 24 24 34? Q Okay. I'll ask the question again, because I

18 (Pages 66 - 69)

25

MR. PLUTA: Object to form.

25 think I didn't get an answer to my question.

Page 70 Page 72 1 THE WITNESS: Where does it say that? 1 A For what he saw then. But a POSITA looking 2 BY MR. TUMINARO: 2 at that technology will not say that I don't have Q Well, Rich explicitly discloses the vertex 3 enough memory to put the vertexes in there, because data by itself, without even considering the pixel 4 the design is consistent with being able to do 5 data; the vertex data is too big to fit in memory 5 vertex processing. 34. right? Q Well, didn't you opine that a POSITA would 7 A Can you refer me to what section of the Rich 7 actually read Rich to interpret that the vertex data you're talking about? is stored in external memory? Q How about I'll refer you to your own A POSITA was doing that around early 2003. declaration at page -- paragraph 33. 10 That's almost seven years or six years after this 11 A Paragraph 33? 11 came out. I would -- I would submit to you that the 12 Q Paragraph 233. 12 memory technology had improved probably by -- by at 13 A Okay. 13 least 30, 40 percent. So I -- I don't see that as a 14 Q And the second sentence reads -- and I'll 14 problem. 15 read it for the record: 15 Q In paragraph 233 of your declaration, did you "For example, Rich discloses that 16 16 or did you not opine that Rich discloses that vertex 17 the user of" -data is stored in external memory, not processor 18 I think that should be "use," not user; is memory 34? 19 that right? A I'm repeating what Rich says, right? I am 20 A Okay. 20 quoting what Rich says. I did not -- it says 21 Q So I'll read it again --21 basically -- referring to column 16, 52 through 55, 22 A Yes. 22 which is consistent with what Rich disclosed and 23 O -- with the correction: what was the understanding at that time. 24 "For example, Rich discloses that 24 What I'm trying to explain to you is that 25 the use of external memory for the 25 five years later, it would have been definitely Page 71 Page 73 1 storage of data for processing is 1 clear to the designers to not be limited but at 2 necessary because the processing 2 local memory. It would have been -- one of the 3 elements 32 only have a small amount 3 first things that would improve is the size of the 4 of dedicated memory." 4 memory. So I would not find that as a problem. 5 Q All right. That wasn't my question on Do you see that? 6 A Yeah. 6 whether it was a problem. Let's just see if we 7 agree that Rich teaches that vertex data is stored Q And the next sentence reads: 8 in external memory, not in the local memory 34. Do "Further, transformed vertex data 9 we agree on that? is stored in external memory after 10 the geometry function." 10 A That's what he says directly in his -- in 11 Do you see that? 11 his -- in his invention, yes. 12 A Yeah, but it -- but --12 O Okay. 13 13 A But -- but if I could qualify that response O There's --14 A -- I think --14 by saying that it would have been very clear to a 15 Q Okay. 15 designer to not be limited by that, because the A -- adding more memory to the processing 16 design is so versatile. So you could do vertex and 16 element 32, it's a minor change, extremely trivial. 17 pixel at the same time. And the technology slope for that -- when was this 18 Q Okay. And isn't it a fact that the pixel 18 disclosed? 1998. That's not an issue anymore for data is larger than vertex data? 20 us. 20 MR. PLUTA: Object to form. 21 21 Q Okay. But --THE WITNESS: It's -- it's hard to say, 2.2. A This is almost 20 years ago. 22 depending on what format you have in mind. But it Q But it's true that on chip memory, because 23 could be -- because you have more data for the --24 that is so small, Rich teaches that it's necessary for the triangle, there are more pixels than

19 (Pages 70 - 73)

vertices. But then vertices are -- you know,

25 to store the vertex data in memory, external memory? 25

Page 74 Page 76 1 they're described in 32-bit or 64-bit floating point 1 on processing elements 32 themselves." 2 2 descriptions. So it depends; the size of the Do you see that? 3 triangle, how many RGB bits you have for the pixel, 3 A Yeah. It's okay. It's -- it's good. You can keep larger data outside the processor. We do 4 so it depends. 4 5 BY MR. TUMINARO: it all the time. Q Let's look at what you said in paragraph 235 6 Q So --7 of your declaration. 7 A What -- what is the -- what -- the -- the 8 A Right. concern you have is -- I'm a little -- you know, I 9 Q "That is, the data size of pixels mean, I don't know what -- what concerns you, but I 10 after rasterization is more than the 10 answer your question, because that's what I have to 11 data size of transformed vertices 11 do, is that if you have larger sized data, you leave 12 12 it outside the processor and you bring it in. after vertex operations." 13 A Okay. 13 That's how systems work. Whether it's the vertex 14 Q Do you see that? 14 data or it's pixel data, it's really irrelevant. 15 A Yeah. 15 So if he doesn't say it, that's how it would Q So you said that, right? 16 16 be done if the internal memory was large enough to 17 A I said that and I agree with that, but it 17 hold the data. 18 depends on the size of the triangles and the type of 18 So I don't see any inconsistency with what 19 pixels you have. 19 I've said here, what is disclosed here or what an 20 If you use a 2-bit RGB, it will not be the 20 engineer could figure out in 2003. 21 21 Q But just so that we're clear, Rich explicitly 22 Q So now, we agree that Rich discloses that 22 discloses that vertex data is stored in external 23 23 vertex data is stored in external memory, correct? memory, not local memory 34? 24 A Rich does that, yeah. 24 A The pixel data? 25 Q Okay. And we agree that pixel data is bigger 25 Q Vertex data. Page 75 Page 77 1 than vertex data, correct? 1 A Vertex data. Yes, he says that. 2 MR. PLUTA: Object to form. 2 Q And you opine that it would be obvious to 3 THE WITNESS: From what -- what it 3 modify Rich to store pixel data in external memory, 4 could be considered that way, yes. 4 as well, correct? 5 BY MR. TUMINARO: 5 A Correct. 6 Q Okay. So then why would you store both 6 Q Okay. Yet you still think it's obvious to vertex data and pixel data in memory 34 and not in 7 7 store both of those things in memory 34 on chip? A The job of memory 34 chip is to provide external memory? 8 9 A The answer is obvious, because you need the executable data to the ALUs. It doesn't matter what 10 data to be closer to the ALU. 10 it is, it's pixel, vertices, whatever you have, that Q Well, in fact, didn't you actually opine that 11 11 is -- the ALU wants to see things close to the ALU. 12 it would be obvious in view of Rich to store pixel 12 So if you have vertex data in the outside memory, in 13 data in external memory? the larger memory, you have to kind of mosey them 14 A Could you show me where I opine that? down to the -- to the local memory or registers, 15 Q Paragraph 234: 15 what have you. "It would have been obvious to one 16 So that is the standard practice. That is 16 17 of ordinary skill in the art that the standard practice. That's what a POSITA would 18 pixel data can also be stored in 18 know, that a processing element would need its 19 external memory after rasterization operations very close to it, whether it's a vertex 20 and pixel processing" -- "and before 20 or pixel. So that's -- that is inconsistent with

20 (Pages 74 - 77)

the knowledge of a POSITA in 2003 time frame and

22 what has been disclosed and what I am saying in this

Q Let's look at paragraph 41 of your

24

25 declaration.

21

22

23

24

25

same" --

A Yeah.

pixel processing because of the

Q -- "concern emphasized by Rich about

the small amount of dedicated memory

	Page 78		Page 80
1	A 41.	1	
2	Q Paragraph 41 reads:	2	whether a graphics standard required a unified
3	"I believe that a person of	3	architecture?
4	ordinary skill in the art relating	4	A Could
5	to the 871 Patent would be someone	5	MR. PLUTA: Object to form.
6	with a good working knowledge of	6	THE WITNESS: Could be, yeah, depending on
7	computer graphic processing	7	what the job requirement was. You could be
8	architecture, as well as the systems	8	designing a piece of the graphics engine that would
9	and programs that support such	9	require you limited amount of knowledge. So it's
10	architecture. The person would also	10	really depends on what you're doing.
11	be familiar with graphics standards	11	BY MR. TUMINARO:
12	as well as general data processing	12	Q Turn to paragraph 19 of your declaration.
13	architecture and techniques."	13	Paragraph 19 reads:
14	Do you see that?	14	"I also understand that the
15	A Yes.	15	relevant inquiry into obviousness
16	Q It goes on to read:	16	requires consideration of four
17	"The person would have gained this	17	factors."
18	knowledge through a Master's Degree	18	Do you see that?
19	in Electrical or Computer	19	A I do.
20	Engineering, or equivalent thereof,	20	Q And the fourth factor that's listed there is
21	and 2+ years of practical working	21	objective factors, correct?
22	experience in the relevant field."	22	A They are what factors?
23	Do you see that?	23	Q Objective factors.
24	A Yes.	24 25	A As opposed to?
25	Q Are you a person of ordinary skill in the art	23	Q That's what's written here
1	Page 79 with respect to your own definition?	1	Page 81 A Okay.
2	A Yes, Counsel.	2	Q in your declaration:
3	Q Okay. What do you mean when you say that a	3	"Objective factors indicating
4	person would be familiar with graphics standards as	4	obviousness or non-obviousness."
5	well as general data processing architecture and	5	Do you see that?
6	techniques?	6	A Yeah. Yes, go ahead.
7	A I think it speaks for itself.	7	Q Did you consider, in forming your opinions
8	Q Well, what does what does it mean to be		
		8	
9		8 9	expressed in your declaration, any objective factors indicating obviousness or nonobviousness?
10	familiar? What are you referring to? A Understand them.		expressed in your declaration, any objective factors indicating obviousness or nonobviousness?
	familiar? What are you referring to?	9	expressed in your declaration, any objective factors
10	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the	9	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have
10 11	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the	9 10 11	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here.
10 11 12	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the	9 10 11 12 13	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors?
10 11 12 13 14	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand.	9 10 11 12 13	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them?
10 11 12 13 14 15 16	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What	9 10 11 12 13 14	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you
10 11 12 13 14 15 16 17	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards?	9 10 11 12 13 14 15 16 17	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your
10 11 12 13 14 15 16 17 18	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003.	9 10 11 12 13 14 15 16 17 18	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis?
10 11 12 13 14 15 16 17 18 19	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003. Q So that's not in, like, the current graphics	9 10 11 12 13 14 15 16 17 18 19	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis? A Discuss
10 11 12 13 14 15 16 17 18 19 20	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003. Q So that's not in, like, the current graphics standards. It's only the 2003 graphics standards?	9 10 11 12 13 14 15 16 17 18 19 20	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis? A Discuss MR. PLUTA: Object to form.
10 11 12 13 14 15 16 17 18 19 20 21	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003. Q So that's not in, like, the current graphics standards. It's only the 2003 graphics standards? A Well, I mean I mean, if he has been	9 10 11 12 13 14 15 16 17 18 19 20 21	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis? A Discuss MR. PLUTA: Object to form. THE WITNESS: Discuss it with who?
10 11 12 13 14 15 16 17 18 19 20 21 22	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003. Q So that's not in, like, the current graphics standards. It's only the 2003 graphics standards? A Well, I mean I mean, if he has been working or she has been working in the field,	9 10 11 12 13 14 15 16 17 18 19 20 21 22	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis? A Discuss MR. PLUTA: Object to form. THE WITNESS: Discuss it with who? BY MR. TUMINARO:
10 11 12 13 14 15 16 17 18 19 20 21 22 23	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003. Q So that's not in, like, the current graphics standards. It's only the 2003 graphics standards? A Well, I mean I mean, if he has been working or she has been working in the field, will will continue if the job required, of	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis? A Discuss MR. PLUTA: Object to form. THE WITNESS: Discuss it with who? BY MR. TUMINARO: Q Do you have any discussion in your
10 11 12 13 14 15 16 17 18 19 20 21 22	familiar? What are you referring to? A Understand them. Q Understand them. Understand all the intricacies of them? I'm trying to understand the scope of what they would understand. A Whatever you need to understand, whatever the job duty is, you have to understand. Q What time frame are you referring to? What graphics standards? A Early 2003. Q So that's not in, like, the current graphics standards. It's only the 2003 graphics standards? A Well, I mean I mean, if he has been working or she has been working in the field, will will continue if the job required, of	9 10 11 12 13 14 15 16 17 18 19 20 21 22	expressed in your declaration, any objective factors indicating obviousness or nonobviousness? A I I considered these factors that I have mentioned here. Q Which factors? A The four factors, scope and content of the prior art do you want me to read that, all four of them? Q Well, in your declaration, did did you discuss any of the objective factors in your analysis? A Discuss MR. PLUTA: Object to form. THE WITNESS: Discuss it with who? BY MR. TUMINARO:

21 (Pages 78 - 81)

Page 82 Page 84 1 obviousness factors are, and in my analysis, I 1 not put in my report; that's for sure. Q So discussion of objective factors apart from 2 included whatever was relevant. Q Well, apart from what appears in -- in 3 paragraph 19 doesn't appear anywhere in your report, paragraph 19, do you discuss objective factors 4 correct? 5 anywhere else in your declaration? 5 A Yeah. If you find it, let me know. A I can -- I mean, let's go through them one by 6 (Exhibit 8 was marked for identification 7 one. It seems to be that would be easier for me if 7 by the court reporter and is attached hereto.) we break it down into -- than just a general 8 BY MR. TUMINARO: question. The differences between the prior art and 9 Q Sir, you've been handed what's been marked as 10 the claim at issue, sure, I've done that. I've done 10 Exhibit 8 for identification purposes. This is U.S. that in my -- in my analysis. We can go through patent number 7,015,913 to Lindholm. 11 12 Do you see that? 12 that. 13 The knowledge of a person of ordinary skill 13 A Yes. 14 in the pertinent art, we just went through that. So 14 Q You considered -- I'm going to refer to this 15 I don't understand -- you want me to identify every 15 as the Lindholm patent; is that fair? 16 one of these factors? 16 A That is fair. 17 17 Q And you considered the Lindholm patent when Q No, that was not my question. 18 A Then I don't understand your question. 18 forming the opinions expressed in your declarations, 19 Q Okay. You say that there are four factors 19 correct? 20 that must be considered in an obviousness analysis, 20 A Yes. 21 21 correct? That's what you say in paragraph 19. Q Okay. And if you turn to page 32 of your 22 A That's right. declaration, there's a section that reads "Ground 23 23 1." Q Okay. And there's -- A is: 24 "The scope and content of the 24 Do you see that? 25 prior art." 25 A Yeah, yes. Page 83 Page 85 1 Correct? Q So just so we're clear, it's your opinion 2 A Uh-huh. 2 that Lindholm anticipates certain claims of the 871 3 3 patent, correct? Q B is: 4 "The differences between the prior A Correct. 5 art and the claims at issue." 5 Q Okay. 6 Correct? 6 MR. PLUTA: I'm sorry. Which declaration 7 7 A Uh-huh, yes. were you referring to? 8 8 MR. TUMINARO: Oh, I'm sorry. I was Q C is: 9 "The knowledge of a person of referring to the -- what's Exhibit 1 with respect to 10 ordinary skill in the pertinent 10 the 871 patent. art." 11 11 MR. PLUTA: Thank you. 12 Correct? 12 BY MR. TUMINARO: 13 A Correct. 13 Q And if you look at -- starting on page 32 of 14 Q And D lists: 14 your declaration, Exhibit 1, with respect to the 871 "Objective factors indicating patent, there is a section that is -- has a heading 15 obviousness or nonobviousness." 16 labeled "Lindholm Discloses Claim 1." 16 17 17 A Correct. Do you see that? 18 Q Okay. Now, my question is -- I'm not 18 A Yes, I do. 19 concerned about the first three factors. I want to 19 Q And that spans all the way to page 34, where, 20 know about objective factors. 20 then, there's a claim chart --21 A Okay. I get that. 21 A I see that. Q Did you discuss objective factors anywhere in 22 22 Q -- correct? Okay. your declaration apart from paragraph 19? 23 And on page 33 of your declaration, you 24 24 reproduce Figure 4 from Lindholm, correct? A I considered them, but I did not report them. 25 I mean, it was -- it was definitely something I did 25 A Yes, I do, correct.

22 (Pages 82 - 85)

Page 86 Page 88 1 shader of claim 1. Q So if you would go to Figure 4 of Lindholm 2 for me and circle in Lindholm Figure 4 what you say 2 A The shader is the -- the processor that is 3 corresponds to the claimed elements in claim 1 of doing the job in general, right? So that's the -the execution units. 4 the 871 patent. 5 Q Let -- let me ask again. In paragraph 90, 5 MR. PLUTA: Object to form. THE WITNESS: You want in Figure 4 to 6 you say: 7 "Components of Execution Pipeline 7 identify what's mentioned in claim 1, those components; is that what the question is? 8 240 correspond to the 'shader' of 9 BY MR. TUMINARO: 9 claim 1." 10 Correct? 10 Q Claim 1 of the 871 patent, yes. 11 A Right, the components of 240. And it's shown 11 A I would refer you to my chart. I think 12 here, Figure 4, and all the components are there. 12 that's more accurate. If you want, you can go 13 through it one by one. I -- I think it would be 13 Q So now my question is, which components of 14 easier for me to explain. I would rather do that. 14 execution 240 correspond to the shader claimed in 15 claim 1? 15 Q Well, I'm asking you if you can mark up 16 Figure 4 of Lindholm and show me where in Lindholm 16 A The shader in claim 1 is identified as 17 you're saying -- what in Lindholm Figure 4 17 Box 62, correct? And that particular function is 18 corresponds to the elements claimed in claim 1 of 18 handled by the execution units. 19 the 871 patent. 19 Q So is it -- is it your testimony -- I just 20 want to make sure I understand -- that execution 20 A I would rather read it from the report, which 21 I think is very clear, in my opinion. 21 unit 470 corresponds to shader -- the claimed 22 shader? 22 So the Lindholm thread control buffer -- so 23 23 claim 1 of 871 -- so claim 1 of 871 basically says A Including all the components that goes with 24 an arbiter -- it's a graphic processing -- the 24 it in terms of the instruction, the sequencers and 25 so on. 25 Lindholm talks about graphic processing in the Page 87 Page 89 1 abstract. Q So that's -- that's my question. I want to 2 Then the -- the next part or the -- of the know what your opinion is. In -- in your -- in paragraph 90, you say: 3 claim 1 talks about an arbiter circuit for selecting 3 one of plurality of inputs in response to a control 4 "Components of Execution Pipeline 5 signal. And in this case, we have the TCB, which is 240 correspond to the 'shader' of working as an arbiter here, deciding between the 6 claim 1." inputs, 215 and 220. These are corresponding to 7 I want to know what components of execution vertex and pixel input data. pipeline 240 you're talking about. 9 Q All right. Let me try this, sir. If you Can you identify them for me in Lindholm? 10 10 would turn with me to paragraph 90 of your A Sure. I say it in paragraph 95: declaration. 11 11 "The Execution Unit 470 eventually 12 12 A Yes. processes the retrieved samples 13 13 Paragraph 90 reads: along with received instructions 14 14 "Because Lindholm's Execution from Instruction Dispatcher 440 'to 15 15 Pipeline 240 performs both vertex perform operations such as linear operations such as transforming 16 interpolation, derivative 16 17 vertices and pixel operations such 17 calculation, blending... and output 18 as texture mapping and blending, 18 the processed sample to a 19 components of the Pipeline 240 19 destination specified by" -- "by 20 correspond to the 'shader' of claim 20 the instruction." 21 21 So that should give you the answer, 22 22 Do you see that? paragraph 95. 23 23 Q So is the answer to my question, then, that A Yes. 24 24 instruction dispatcher 240 and execution unit 470 Q So what I want to know is what components of 25 execut- -- execution pipeline 240 correspond to the correspond to the claimed shader --

23 (Pages 86 - 89)

Page 90 Page 92 1 MR. PLUTA: Object to form. 1 the shader and perform the operation. So that's --2 BY MR. TUMINARO: 2 to me, that's clear, at least from what I explained. 3 Q -- of --3 Q I guess it's -- it's not clear to me. I 4 don't know -- is it 470 and instruction dispatcher 4 MR. PLUTA: I'm sorry. 5 BY MR. TUMINARO: 5 240 from Lindholm that make up the -- the claimed Q -- the 871 patent? shader, or are there other things that you are 7 MR. PLUTA: Object to form. pointing to in Lindholm that correspond to the 8 THE WITNESS: There are -- there are 8 shader? 9 components that are related to the shader that help 9 A And the define -- the definition of the 10 470 to execute. So I -- I don't know where you want shader is? You -- the definition that is in 871? 11 O The claimed shader in 871. 11 to put the -- the border in terms of where the 12 shader is. 12 A The claimed shader in 871. Well, it shows 13 But the shader, if you want to count it as 13 the unified shader in this figure, right, if you're 14 the execution unit, it would be the 470. The referring to that. Assuming what's put into that 15 instructions are coming from above, as you see in 15 871 -- Figure 5. Hold on a second. 16 Figure 4. 16 Okay. So the -- looking at 871, the unified 17 BY MR. TUMINARO: 17 shader is defined to be including what's in here, 18 Q It's not where I want to put the borders. I 18 Figure 5, correct? 19 want to know where you put the borders. I want to 19 So that means it includes the instruction 20 know what your opinion is. Which components -store. So I would put anything to do with 21 A And I -instructions as part of the shader based on what 22 Q -- are -- are corresponding to the shader? 22 this -- these guys have defined in 871. 23 23 A The shader is --Q Okay. 24 MR. PLUTA: Object to form. 24 A And that is consistent with this 25 THE WITNESS: -- defined as the execution 25 understanding. Page 93 1 unit. I've -- I've said it several times -- I mean, Q So what structures, then, are you talking 1 2 I've repeated my answer several times. The 2 about that correspond to the instructions -- strike 3 that. 3 execution unit 470 processes the received samples 4 along with the received instructions from 4 What structure in Lindholm are you referring 5 to that corresponds to the instruction store? 5 instruction patch -- instruction dispatch 440. So 6 instruction dispatch provides the instructions to 6 A Anywhere you see an instruction, it's 470 to do the computation. referring to this information that is in Figure 5. It says "instruction store." They -- they BY MR. TUMINARO: Q Okay. So it's -- 440 and 470 correspond to identified this figure to be the shader based on their definition of a shader. And that's consistent 10 the unit -- to the shader claimed in claim 1. with this figure. So that includes anything to do 11 Is there anything else that you're pointing 12 to that corresponds to the shader in claim 1? with the instruction, Counsel. 13 Q Sir -- sir, I --13 MR. PLUTA: Object to form. 14 MR. TUMINARO: I'll ask again. 14 A Let me finish my -- you're not -- you're not 15 letting me finish. 15 Q Is there anything else in Lindholm besides 16 instruction dispatcher 440 and execution --Q I didn't say anything. 16 A You started talking, sir. execution unit 470? Is there anything else besides 17 18 those two components that you're saying corresponds 18 But whatever has to do with instruction is to the claimed shader in claim 1 of the 871 patent? part of this shader.

24 (Pages 90 - 93)

Q That's -- that's what I want to know. What

things in -- in Figure 4 or in Lindholm generally are you pointing to that correspond to the claimed

shader? You said the execution unit 470, the

Is there anything else in Lindholm that

instruction dispatcher 440.

20

23

24

25

20

24

23 computation.

A The -- the way that I interpret your question

But there are other components that feed into

21 is that the shader is the execution unit part of the

25 this block. So there are instructions that come to

22 computation, and that's the one 470 in terms of the

Page 96

- 1 you're pointing to that corresponds to the claimed
- 2 shader?
- 3 A So --
- MR. PLUTA: Objection; asked and answered. 4
- 5 THE WITNESS: It -- it includes instructions,
- 6 it includes the register file, it includes anything
- to do with what has been defined in 871 to be the
- shader, consistent with that.
- 9 BY MR. TUMINARO:
- Q So -- just so I understand now, register file 10
- 11 450 in Lindholm you're saying corresponds to the
- 12 claimed shader, as well?
- 13 A Because there are source registers here for
- 14 the CPU based on Figure 5 of -- of the 871, so there
- 15 are registers corresponding to this.
- 16 Q Okay.
- 17 A If -- if you open up execution unit 470, if
- 18 there are registers in there, that will be
- corresponding to the -- to the Figure 5.
- 20 Q Okay. Anything else in -- in Lindholm,
- 21 besides instruction dispatcher 440, execution unit
- 22 470, register file 450, anything else in Lindholm
- 23 that you're saying corresponds to the claimed shader
- 24 in 871?
- 25 MR. PLUTA: Object to form.

- 1 as defined here for the unified shader. I'm
- 2 identifying -- that's what you wanted me to do.
- 3 Q Right.

Page 94

- 4 A I've identified those pieces that match
- 5 Figure 5 here.
- Q Okay. So Figure 5 is defining the invention
- 7 or the -- it's describing what's in the 871 patent,
- right, not what's in Lindholm, right?
- 9 A No. I was using that as a model to explain
- 10 to you what the unified shader defined here
- 11 corresponds to here. That's what you wanted me to 12 do.
- 13 Q Right. And it's your opinion that Lindholm
- 14 discloses what's in the -- claim 1 of the 871
- 15 patent, right?
- A Based on Figure 5, yes. 16
- 17 Q Okay. So what I'm trying to understand is
- 18 the scope of your opinion. I want to know what --
- if there's anything else in Lindholm besides
- instruction dispatcher 440, register file 450,
- execution unit 470, is there anything else that
- you're pointing to in Lindholm that corresponds to
- the claimed shader, in your opinion? I want to know
- 24 what your opinion is.
- 25 MR. PLUTA: Object to form.

Page 95

- THE WITNESS: Whatever 871 -- 871 defines in 1
- 2 Figure 5 to be the unified shader, they have coined
- 3 that for their -- for their understanding of unified
- 4 shader. It may be different from NVIDIA. It might
- 5 be different from somebody else.
- So referring to Figure 5, referring to Figure 6
- 7 5 -- I'm going to repeat what's in Figure 5. There
- 8 are instruction stores. There's a -- there's a
- constant, which are -- these are specialized
- 10 functions. There are source register operands.
- 11 There is a CPU which does the computation, and there
- 12 is block 92. And that's consistent what's in that
- 13 page.
- 14 (Unintelligible reading) to be
- 15 process (unintelligible reading) to the general
- 16 purpose register block 92. So that's -- that's --
- 17 if they define that to be the part of the shader,
- 18 and it's right here, the register file.
- 19 BY MR. TUMINARO:
- 20 Q Okay. Just -- we'll agree that the 871
- 21 Figure 5 is not referring to Lindholm, right?
- 22 A Excuse me?
- 23 Q Figure 5 of the 871 patent is not referring
- 24 to Lindholm, right?
- A I -- no. I'm just trying to explain to you

Page 97

- 1 THE WITNESS: They did not explain the
- details about how they were --
- 3 BY MR. TUMINARO:
- Q Who's they?
- 5 A 871.
- 6 O Okay.
- 7 A There are additional information -- there's
- just one Figure 5 to show the exploded
- description -- they use the word "exploded"
- description of the shader. This is not sufficient
- to explain whether the score boarding algorithms are
- 12 in there, whether the other components are there.
- 13 Because it talks about resources not being
- 14 available.
- 15 If you read 871, which I'm sure you have, it
- 16 says we will switch from vertices to pixels,
- depending on resources being available. There is
- 18 nothing here to tell me how that switching is done.
- 19 But -- but Lindholm has a lot more detail in
- terms of explaining what's going on. So given what's in Figure 5, I have identified what's
- 22 available in Lindholm, but there are other things in
- Lindholm that 871 did not discuss.
- 24 MR. TUMINARO: I'd like to take a break, I
- 25 guess, to -- to change the tape, and we'll come

25 (Pages 94 - 97)

Page 98 Page 100 1 back. 1 Exhibit 2, your declaration, number 330. 2 THE VIDEOGRAPHER: This -- we are off the 2 A Okay. 3 record. The time is 11:46 a.m., and this is the end 3 Q Are you there? 4 A Yes. of the first media. 5 Q All right. I just want to confirm, I 5 (Lunch recess.) THE VIDEOGRAPHER: This is the beginning of 6 misspoke earlier. You did not actually consider Kizhepat in forming the opinions expressed in the 7 the second media. We're back on the record. The 8 time is 12:37 p.m. declaration for the 369 patent, which is Exhibit 2? 9 A Yeah. Please continue. 10 BY MR. TUMINARO: 10 Q Right. Okay. 11 Thank you, Counsel. 11 Q Welcome back, sir. 12 All right. Now -- so we read the field of 12 A Good afternoon. 13 Q During the break, did you talk with your 13 the invention for Kizhepat. 14 counsel about the substance of your testimony? 14 A Kizhepat. 15 Q Yeah, okay. It's a fact, is it not, that the 15 A No. 16 MR. TUMINARO: Okay. 16 acronym GPU appears nowhere in Kizhepat? 17 17 Did you give him the exhibit? A Okay. I have to check. 18 THE REPORTER: Yes. 18 So it's mostly talking about data processing 19 (Exhibit 9 was marked for identification 19 system. By looking at it very quickly, I did not see the word "GPU," but data processing of this type 20 by the court reporter and is attached hereto.) of architecture is consistent with the understanding 21 BY MR. TUMINARO: of doing graphics operations. 22 Q You have in front of you Exhibit 9. It's 23 23 U.S. patent number 7,376,811. This is the Kizhepat Q And the fact is Kizhepat doesn't use the 24 reference. 24 phrase "graphics processor," correct? 25 A It's -- it -- it is data processing, 25 A Kizhepat. Page 99 Page 101 Q Yes? 1 1 and some aspects of graphic processing is data A Yes, I have it. processing. 3 Q And you considered Exhibit 9 in forming the 3 Q Well, answer me this. Is the word -- does 4 opinions expressed in both of your declarations? 4 the word "graphics" appear anywhere in Kizhepat? 5 A I believe so. A I looked at it quickly, and I did not come Q Okay. All right. If we look at the "Field 6 across it, but that doesn't mean I looked at every 6 7 of the Invention," it's in column 1. It reads: 7 single sentence. 8 "The present invention relates to Q Well, look at every single sentence and tell 9 system architectures for data 9 me, does the word "graphics" appears anywhere in 10 10 Kizhepat? processing, and more particularly to 11 an architecture based upon a A Okay. So all the claims refer to data 12 hardware engine which performs 12 processing. I looked at the claims. I think, 13 operations and computations on data 13 judging from 871, which is the preamble of claim 1, 14 as the data traverses paths 14 and I think Lindholm also talks about graphics 15 controlled by software." 15 processing. I just want to make sure that's the 16 16 case. Do you see that? 17 A Column 1, "Field of Invention," right? 17 Yeah, so it's not in the claim description. 18 18 "Traverses paths controlled by software," yes. You Q So we agree that Kizhepat neither uses the acronym "GPU" nor the word "graphics" anywhere --19 read the paragraph correctly, yes. 20 20 Q I apologize. In the -- the declaration that A In the claim disclosures, right. 21 you submitted for the 330 IPR, Exhibit Number 2, you 21 Q In fact, it doesn't appear -- in addition to 22 the claims, it doesn't appear anywhere in the 22 did not consider the Kizhepat reference for that 23 declaration, right? If you look at the 23 patent? 24 24 materials considered -- strike that. MR. PLUTA: Object to form. 25 Just look at the materials considered in 25 THE WITNESS: I would assume that if they had

26 (Pages 98 - 101)

	Page 102		Page 104
1	mentioned it, it would be in the description	1	(Exhibit 10 was marked for identification
	specification, as well, so	2	by the court reporter and is attached hereto.)
3		3	MR. TUMINARO: This is 10, right?
4	, ,	4 THE REPORTER: 10.	
5		5	BY MR. TUMINARO:
6	*	6 Q You've been handed what's been marke	
7	U 1 ,	7 Exhibit 10 for identification purposes. This	
8	A I was looking at that. And what I used	8	this is the decision by the PTAB regarding
9		9	institution of the IPR2015-00326, correct? A Correct.
10		11	
12	THE REPORTER: "Regarding" what? THE WITNESS: Multiplexers.	12	Q And you've seen this document before? A I have.
13	And the fact that you're able to select one	13	
14	•	14	Q Okay. So I'd like to direct your attention to page 9 at the bottom, the last sentence that
15	So that that's basically the level of	15	that bridges across page 9 and into page 10. I'll
16	•	16	read it for the record:
17	• 1 • 1	17	"Thus, for purposes of this
18	"A typical and common multiplexing system is described in Kizhepat.	18	decision, we construe the 'means for
19	Kizhepat discloses a hardware engine	19	performing vertex operations and
20	for data processing that includes a	20	pixel operations and performing one
21	plurality of functional units and	21	of the vertex operations or pixel
22	data routing units that interconnect	22	operations based on the selected one
23	the functional units."	23	of the plurality of inputs' to
24	So that's been	24	include a register, an instruction
	BY MR. TUMINARO:	25	sequencer capable of providing
1	Page 103 Q Okay. I appreciate I appreciate the	1	Page 105 instructions for performing vertex
2		2	operations and pixel operations, and
3	My question was, in your declaration, when	3	a processor capable of floating
4	you're talking about Kizhepat, you didn't cite	4	point, arithmetic, and logical
5	anywhere in Kizhepat just that Kizhepat discloses	5	operations on a selected input."
	the word "GPU" or "graphics processor," right?	6	Do you see that?
7	A It's not in my report.	7	A Yes.
8	Q Okay. Switching gears to the materials	8	Q Did I read it correctly?
9		9	A I didn't check.
	decision with respect to the 871 patent?	10	Q Okay. My question is, do you agree with the
11	A I think you asked that this morning, right?	11	board's construction for this means for limitation?
12	Q Well, I asked if you reviewed anything else.	12	A My understanding is that we have to accept
13		13	the board's understanding or construction.
14		14	That's my understanding from legal point of view.
15	A You did ask, and I said no.	15	I don't know if there is and I agree with
16	Q So are you aware of the fact that the board	16	
17		17	"To include the register,
18	A I'm sorry. You asked me about the response	18	instruction sequencer capable of"
19	you gave to I apologize. You asked me about the	19	"we construe the means of performing
20	response you gave for some other case.	20	vertex operations and pixel
21	Did I look at the review did I look at the	21	operations and" (unintelligible
22		22	reading)
23	MR. TUMINARO: Okay. Maybe this will make it	23	THE REPORTER: I'm sorry.
	faster.	24	THE WITNESS: I'm sorry. I'm just I
25	Can you grab that.	25	should read it for myself.
1		1	-

27 (Pages 102 - 105)

Page 106 Page 108 1 It's identifying what they say should be 1 between the inputs. A more sophisticated control 2 included. I would say that plus additional stuff, 2 will be the -- the logic behind that control signal. 3 if you want to get to the details. That's the 3 BY MR. TUMINARO: 4 minimum requirement, I would say. Q In the past, you've opined that an arbiter 5 BY MR. TUMINARO: receives and provides, correct? Q Are you -- are you saying that the board MR. PLUTA: Object to form. Object to 6 7 relevance. 7 missed out some structure and there should be additional stuff in their construction? THE WITNESS: In the past meaning what? What 9 A No, I'm not saying that. I'm saying these do you mean in the past? 10 are the basic components. And depending what else 10 BY MR. TUMINARO: you want to do, you may add additional stuff to it. O In --11 12 It de- -- it depends on the level of -- so I -- it's 12 A In my history of my life? 13 consistent with my understanding. Q -- in another -- in -- in another proceeding 14 O Okay. You've heard of the word "arbiter" 14 you have opined that arbiter means -- refers to 15 before? structure that receives and provides, correct? 16 A In the context of? 16 MR. PLUTA: Object to form. Object to 17 Q Graphics processing. 17 relevance. Counsel, it's a different patent, 18 A Yes. 18 different references, different terms. 19 Q Okay. And an arbiter selects from available 19 THE WITNESS: I just explained to you what a 20 inputs? 20 MUX is. It has two inputs and has outputs in this 21 A Correct. 21 case, so it's a two-to-one selector. 22 Q Okay. If you'd turn with me to the 871 22 BY MR. TUMINARO: 23 patent, it's Exhibit 7. Q Is it or is it not true that in the past, in 24 A 87- -- 871. What page? 24 another proceeding, you opined that arbiter is 25 Q 871 patent, Figure 4A. 25 structure that receives and provides? Page 107 Page 109 1 A 4A. 1 MR. PLUTA: Object to form. Same objection 2 Q Are you there, 4A? as to relevance. 3 3 A Yep, yes. THE WITNESS: I -- I don't know in what Q All right. My question is, you see MUX 66 at context that was, but in general, a MUX has an input 5 the top of Figure 4A? and output. There's no question about it. It's a 6 A I see it. circuit that has an input and output. 7 7 I don't know what -- what -- can you repeat Q Is MUX 66 an arbiter? 8 what you just said about what I --MR. PLUTA: Object to form. 9 THE WITNESS: Does it say it's an arbiter? 9 BY MR. TUMINARO: 10 10 It says it's a MUX. Q Sure. I'll repeat the question again. 11 11 BY MR. TUMINARO: Is it or is it not true that in the past, in 12 Q Well, that's my question. 12 another proceeding, you opined that an arbiter is 13 A Unless it's a typo. 13 structure that receives and provides? 14 Q That's my question. Do you consider, in your 14 MR. PLUTA: Object to form. Object to 15 relevance. 15 opinion, MUX 66 to be an arbiter? MR. PLUTA: Object to form. 16 THE WITNESS: My answer to you is that it 16 17 THE WITNESS: If -- if you want to implement 17 inputs and outputs. Receives an input and generates 18 an arbiter, you need to have a control, and you need 18 an output.

28 (Pages 106 - 109)

20

22.

23

25

21 output?

BY MR. TUMINARO:

A Of course.

24 to what I just said.

Q An arbiter receives an input and generates an

I mean, I -- I don't know what else I can add

Q Going back to your declaration, with respect

to have a way of selecting between inputs. A MUX

does that on its own, and a simple control would

basically let you select between the inputs. You

22 can basically select which one of the inputs would

want, depending on what -- how you control it.

A simple control will be just selecting

23 be the output. You can call that an arbiter if you

20

24

25

	D 110		D 112
1	Page 110 to the 871 patent, Exhibit 1.	1	MR. PLUTA: Object to form.
2	A What page?	2	THE WITNESS: I have it the sequencer,
3	Q I'm trying to get there.		which is identified in Figure 5, it's also discussed
4	Page 32.		here to be the instruction scheduler and dispatcher
5	A Okay.		and the and the components related to that. Then
6	Q Paragraph 91 reads:		the register file and the execution unit. So I've
7	"Lindholm also discloses a Thread	7	
8	Control Buffer 420 which is read as		know what other component you're referring to.
9	the 'arbiter circuit' of claim 1."	9	The sequencer is the one that sequences
10	Do you see that?	10	through the through the steps, and that's what
11	A I see that.	11	what instruction scheduler and dispatcher do.
12	Q So thread control buffer 420 appears in	12	BY MR. TUMINARO:
13	Figure 4 of Lindholm, right?	13	Q I'm still very confused. I apologize.
14	A It does.	14	In claim 1 of the 871 patent, there's claimed
15	Q And that thread control buffer 420 is what	15	a shader, correct?
16	you're saying corresponds to the claimed arbiter	16	A Yes.
17	circuit, correct?	17	Q Okay. In paragraph 90 of your declaration,
18	A Correct.	18	you say:
19	Q Is there anything else in Lindholm that	19	"Components of the Execution
20	you're pointing to or that you opine that	20	Pipeline 240"
21	corresponds to the claimed arbiter circuit?	21	Which you're referring to Figure 4 of
22	A That's just one example.	22	Lindholm. You say that certain:
23	Q The thread what's just one example?	23	"Components of Execution Pipeline
24	A TCB is an arbiter.	24	240 correspond to 'shader' of claim
25	Q TCB 420, you're saying, is an arbiter?	25	1."
	D 111		
	Page 111		Page 113
1	A Right.	1	Page 113 Correct? You said that in paragraph 90?
1 2		1 2	
	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of	2	Correct? You said that in paragraph 90?
2 3 4	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of	2	Correct? You said that in paragraph 90? A I said register file, the processing unit,
2 3 4 5	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes.	2 3 4	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer,
2 3 4 5 6	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that	2 3 4 5 6	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in
2 3 4 5 6 7	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter	2 3 4 5 6 7	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words.
2 3 4 5 6 7 8	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit?	2 3 4 5 6 7 8	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction
2 3 4 5 6 7 8 9	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm	2 3 4 5 6 7 8 9	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher.
2 3 4 5 6 7 8 9 10	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think	2 3 4 5 6 7 8 9	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else?
2 3 4 5 6 7 8 9 10 11	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not	2 3 4 5 6 7 8 9 10 11	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said.
2 3 4 5 6 7 8 9 10 11 12	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay.	2 3 4 5 6 7 8 9 10 11 12	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay.
2 3 4 5 6 7 8 9 10 11 12 13	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else.	2 3 4 5 6 7 8 9 10 11 12 13	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record.
2 3 4 5 6 7 8 9 10 11 12 13 14	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it	2 3 4 5 6 7 8 9 10 11 12 13 14	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record.
2 3 4 5 6 7 8 9 10 11 12 13 14 15	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears	2 3 4 5 6 7 8 9 10 11 12 13 14 15	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.)
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered:	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that goes with it in terms of the	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m. Please continue.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that goes with it in terms of the instruction, the sequencers and so	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m. Please continue. BY MR. TUMINARO:
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that goes with it in terms of the instruction, the sequencers and so on."	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m. Please continue. BY MR. TUMINARO: Q Would you turn back to Exhibit 1 for me.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that goes with it in terms of the instruction, the sequencers and so on." A Right.	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m. Please continue. BY MR. TUMINARO: Q Would you turn back to Exhibit 1 for me. A Yeah.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that goes with it in terms of the instruction, the sequencers and so on." A Right. Q So you is it your opinion that anything in	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m. Please continue. BY MR. TUMINARO: Q Would you turn back to Exhibit 1 for me. A Yeah. Q And paragraph 22, which is on page 11.
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	A Right. Q Right. Okay. My question is so you say that corresponds to the claimed arbiter circuit of claim 1? A Yes. Q Is there any other structure in Lindholm that you're relying on to meet the claimed arbiter circuit? A My position is that's the arbiter that I'm using to discuss claim 1 in my charts. I think that's what I've said. I have not Q Okay. A opined on anything else. Q Okay. All right. Earlier I asked you, is it your opinion that execution unit 470, which appears in Figure 4, corresponds to the claimed shader, and you answered: "Including all the components that goes with it in terms of the instruction, the sequencers and so on." A Right. Q So you is it your opinion that anything in	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	Correct? You said that in paragraph 90? A I said register file, the processing unit, which is the execution unit, and the sequencer, which in this case would be sequencing the execution unit. Q What are you saying is the sequencer in Figure 4 of Lindholm? I don't see those words. A Instruction scheduler, instruction dispatcher. Q Anything else? A That's all I said. MR. TUMINARO: Okay. Could we go off the record. THE VIDEOGRAPHER: We are off the record. The time is 1:04 p.m. (Recess.) THE VIDEOGRAPHER: We're back on the record. The time is 1:27 p.m. Please continue. BY MR. TUMINARO: Q Would you turn back to Exhibit 1 for me. A Yeah.

29 (Pages 110 - 113)

Page 114 Page 116 1 "I have been informed that the 1 illustrated in Figure 3." 2 2 Do you see that? application that issued as the 871 3 patent was filed in November 2003. 3 A Yeah. 4 4 Q Okay. And if you turn to Figure 5, the third As a result, I will assume the 5 line -- are you there? I mean column 5, the third 5 relevant time period for determining line. Excuse me. 6 what one of ordinary skill in the 7 art knew is early to mid 2003." 7 A Uh-huh. 8 Do you see that? 8 Q Column 5 --A Yes. 9 A Yes. 9 Q So you didn't consider what a person of 10 Q -- the third line reads: 11 ordinary skill would have known in, for example, 11 "Figure 3 is a flow chart of geometry processing aspects of the 12 2002, right? 12 13 A It would be not from that statement, but it's 13 image generation system of Figure 14 additive. Whatever they knew in 2002, they have 14 1." 15 15 gained some knowledge since then. So I -- I see it A Okay. 16 like an additive. Q So those two sentences in combination, if you Q Okay. Is there anywhere in your dec- -- in read line 1 of column 9 and the fact that Figure 3 17 18 your declaration where you talk about what a person relates to geometry processing, that means the first of ordinary skill in the art would have known in function carried out by the system is geometry 20 2002? processing as illustrated in Figure 3, correct? 21 21 MR. PLUTA: Object to form. MR. PLUTA: Object to form. 22 THE WITNESS: No, but it is consistent with 22 THE WITNESS: It says it's geometry 23 what I understand, they keep adding to their processing, but it doesn't say that it -- it is 24 knowledge to that point, not all happening in that limited to doing that sequentially with respect to 25 time frame. It would be gradual. pixel. I don't get that --Page 115 Page 117 1 BY MR. TUMINARO: 1 MR. TUMINARO: Okay. 2 Q Okay. Switching gears --2 THE WITNESS: -- information out of that. 3 A Okay. 3 BY MR. TUMINARO: Q -- Rich performs vertex operations and pixel Q Okay. Then let's look at line 40 in column 5 operations sequentially, correct? 5 9. It says: MR. PLUTA: Object to form. 6 "After geometry processing, the 7 BY MR. TUMINARO: 7 next function carried out by the 8 Q Let me ask that again. image generation system is 9 The system disclosed in Rich performs vertex rasterization." 10 10 operations and pixel operations sequentially, Do you see that? correct? 11 11 A Yes. 12 A Let me check. Is there a specific section 12 Q And rasterization is a pixel operation? 13 you want me to refer to, or is this a general A Yes, but --14 question? Do you have a reference so I can look at 14 Q So pixel operations occur after geometry 15 it? 15 processing? 16 Q All right. Look at column 9. A You have to have the primitives before you 17 A Okay. I don't see the word "sequentially can do rasterization, but -- it says -- first of 18 executed." all, in one embodiment, I think it said that as 19 Q All right. Let me help you out, then. much. There is no limitation from what I see here 20 A Yeah. 20 whereby the control unit can activate primitives 21 Q Look at the first sentence in column 9. It 21 being executed into the corresponding rasterization 22 reads: 22 while some other primitive's being evaluated, 23 because you have multiple primitives. So it is not "The first function carried out by 24 one particular embodiment of the limiting that because you have multiple processing 25 image generation system is units, and the control unit can control these

30 (Pages 114 - 117)

Page 118 Page 120 1 independent of each other. 1 correct? 2

9

16

Q I'm asking about what's actually disclosed in

3 Rich. And in column 9, it says first you do

geometry operations, and after the geometry

operations, you do pixel operations; is that right?

A In one particular embodiment, but it doesn't

say that's the only way to do it.

Q Point to me an embodiment where it says they're not done sequentially.

A It didn't need to, because it was obvious to

11 somebody who was looking at this -- or it was clear

12 to somebody who was looking at this that I have

13 multiple processing elements. I have a central

14 control unit that I can control these processing

15 elements.

What I want to do is to able to do a linear 16

17 expression evaluation on a subset of these

18 primitives, convert them into pixels while I'm

working on the primitives of some other portions of

20 the graphics image.

21 So I would say it's -- in other embodiments,

22 that could happen. I don't see any limitations here

23 on that.

24 Q Okay. I'm not asking about what could

25 happen, what -- I want to know, is there anything in

MR. PLUTA: Object to form.

3 THE WITNESS: I haven't checked all the

4 designs at that time frame. There are hundreds of

patents and disclosures and so on. I couldn't tell

you that for a fact, but it is possible that they

7 were there.

8 BY MR. TUMINARO:

Q Well, you're aware of the fact that ATI's

10 Xenos processor is recognized as -- in the industry

as the first to provide a graphics processor with a

unified shader, correct?

13 MR. PLUTA: Object to form, lack of

14 foundation --

15 THE WITNESS: I don't have any --

MR. PLUTA: -- relevance.

17 THE WITNESS: -- indication of that. But I

18 can assure you in conferences and journals, which

they do the foundation work that ends up in these

products, there has been reports of unified shaders

prior to that.

22 BY MR. TUMINARO:

23 Q And you didn't cite any of those reports

anywhere in your declaration that talked about a

unified shader before 2003, correct?

Page 119

1 disclo- -- disclosed in Rich --

2 A The block diagram.

3 Q -- that says that you do geometry operations

4 and pixel operations not sequentially?

MR. PLUTA: Object to form. 5

THE WITNESS: I think the block diagram gives

you that -- that information. But if you want to

8 see it written and happening concurrently, it

9 doesn't say anything about sequentiality either. So

10 I don't have any specific sentence to saying it's

11 happening sequentially or otherwise concurrently.

12 But -- but Figure 2 in this case is

13 indication to me that the -- the operation could

happen in a partition way, namely, portions of the

primitives could be handling -- could be working on

16 rasterization while the other parts are doing vertex

operations. So I -- I would say that is quite

18 possible with this design, although it didn't say it

specifically.

20 BY MR. TUMINARO:

21 Q Okay. And given your knowledge of graphics

22 processing generally, you're aware that there were

no graphics processors --23

24 A Uh-huh.

25 Q -- in 2003 or 2002 that had a unified shader,

Page 121 A I didn't need to. I was convinced with what

2 I had.

1

3 Q If you turn back with me to the institution

decision --

5 A Yeah.

6 O -- in the 326 case.

7 A Yes.

Q On page 22, the second paragraph, the third

sentence of the second paragraph, it starts with

"Nevertheless."

Do you see that? 11

12 A Yes.

15

17

13 O I'll read it:

14 "Nevertheless, as discussed above,

both Petitioner and Patent Owner

16 recognize that Rich discloses

performing vertex operations and

18 pixel operations sequentially."

19 Do you see that?

A Yeah. 20

21 Based on what it --

22 MR. PLUTA: There's no question pending.

23 THE WITNESS: Oh, sorry.

24 BY MR. TUMINARO:

25 Q So during vertex operations, the system

31 (Pages 118 - 121)

Page 124 1 disclosed in Rich will be performing operations on 1 ALU and microprocessors? vertex data, correct? 2 3 MR. PLUTA: Object to the form of the 3 Q Is it your testimony that sometimes the question, lacks foundation. I also object to the terminology of microprocessor and ALU is used relevance because, Counsel, you're reading from a interchangeably? ground that was not instituted by the board in this A Sometimes, yeah. 6 7 7 decision. Q Going to the Rich reference, is it your 8 THE WITNESS: Okay. Thank you. opinion that vertex data and pixel data in Rich can 9 You need an answer? be stored on external memory? 10 BY MR. TUMINARO: 10 A It is, yeah. 11 O I do. Q Is it your opinion that vertex data and pixel A I think I gave you my answer regarding what 12 12 data in Rich may also be stored in an internal 13 Rich can do and cannot do. From understanding of 13 memory? 14 the technology, my position is that one could 14 A That's right. 15 implement that in parallel because it has all the 15 MR. PLUTA: I don't have any ques- -- further 16 resources as defined in the description. 16 questions at this time. 17 MR. TUMINARO: All right. I have no more 17 MR. TUMINARO: I have no more. 18 questions at this time. 18 THE VIDEOGRAPHER: We are off the record. 19 THE WITNESS: Okay. The time is 1:42 p.m., and this concludes today's 20 MR. PLUTA: I have a couple questions. testimony given by Dr. Nader Bagherzadeh. The total 21 Should I -- I'll just -number of media used was two and will be retained by 22 THE WITNESS: Okay. Veritext Legal Solutions. 23 MR. PLUTA: It's a little awkward for the 23 (TIME NOTED: 1:42 p.m.) 24 video, but... 24 25 **EXAMINATION** 25 Page 123 Page 125 1 BY MR. PLUTA: I, the undersigned, a Certified Shorthand 2 Reporter of the State of California, do hereby Q Do you remember the questions regarding your 3 certify: 3 CV earlier about graphics processing? 4 That the foregoing proceedings were taken 5 before me at the time and place herein set forth; 5 Q And you remember -- you listed quite a few that any witnesses in the foregoing proceedings, publications or journals that counsel did not prior to testifying, were placed under oath; that a include that were on your CV related to graphics verbatim record of the proceedings was made by me processing, correct? using machine shorthand which was thereafter 9 A Correct. transcribed under my direction; further, that the 10 MR. TUMINARO: Objection; leading. foregoing is an accurate transcription thereof. 11 THE WITNESS: Correct. 12 I further certify that I am neither 12 BY MR. PLUTA: 13 financially interested in the action nor a relative 13 Q What's your experience with graphics or employee of any attorney of any of the parties. 14 processing outside of just published papers or IN WITNESS WHEREOF, I have this date 15 15 conference papers? 16 subscribed my name. A The work I did at Morpho Technologies 16 17 17 developing the SIMD architecture for pixel Dated: September, 18 2015 18 18 processing. 19 19 THE REPORTER: "Developing" what? 20 20 THE WITNESS: The SIMD processor for pixel 21 21 processing. 22 There are too many processing. 22 CSR No. 8714 23 BY MR. PLUTA: 23 24 Q You remember you were -- you remember you 24 25 were asked some questions about the terminologies of 25

32 (Pages 122 - 125)

	Page 126		Page 128
1	LG Electronics Inc v. ATI Technologies ULC		LG Electronics Inc v. ATI Technologies ULC
2	Dr Nader Bagherzadeh	2	Dr Nader Bagherzadeh
3	INSTRUCTIONS TO THE WITNESS	3	ACKNOWLEDGMENT OF DEPONENT
4	Please read your deposition over	4 I,, do	
5	carefully and make any necessary corrections.	5 hereby certify that I have read the foregoing	
6	You should state the reason in the	6	pages and that the same is a correct
7	appropriate space on the errata sheet for any	7	transcription of the answers given by
8	corrections that are made.	8	me to the questions therein propounded,
9	After doing so, please sign the errata	9	except for the corrections or changes in form
10	sheet and date it.	10	or substance, if any, noted in the attached
11	You are signing same subject to the	11	Errata Sheet.
12	changes you have noted on the errata sheet,	12	
13	which will be attached to your deposition.	13	
14	It is imperative that you return the	14	DATE SIGNATURE
15	original errata sheet to the deposing	15	DITTE SIGNATURE
16	attorney within thirty (30) days of receipt	16	Subscribed and sworn to before me this
17	of the deposition transcript by you. If you	17	day of, 20
18	fail to do so, the deposition transcript may	18	
19	be deemed to be accurate and may be used in	19	My commission expires:
20	court.	20	
21		21	Notary Public
22		22	
23		23	
24		24	
25	2134835	25	2134835
25		25	2134835
	Page 127	25	2134835
1	Page 127 LG Electronics Inc v. ATI Technologies ULC	25	2134835
1 2	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh	25	2134835
1 2 3	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A	25	2134835
1 2 3 4	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A	25	2134835
1 2 3 4 5	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE	25	2134835
1 2 3 4 5 6	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE	25	2134835
1 2 3 4 5 6 7	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE	25	2134835
1 2 3 4 5 6	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE	25	2134835
1 2 3 4 5 6 7 8 9	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	Page 127 LG Electronics Inc v. ATI Technologies ULC Dr Nader Bagherzadeh E R R A T A PAGE LINE CHANGE Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason: Reason:	25	2134835

33 (Pages 126 - 128)

[**& - 64**]

&	15th 7:6	220 87:7	4
& 8:2	16 33:12 72:21	23 25:12	4 5:20 36:7,11,18
0	160 23:13	233 70:12 72:15	85:24 86:1,2,6,16
	17 33:16	234 75:15	86:17 88:12 90:16
053 15:2,4,10,18	18 24:24 125:18	235 74:6	93:21 110:13
1	19 80:12,13 82:4,21	240 87:15,19,25	111:16,24 112:21
1 5:10 10:6,10,16,17	83:23 84:3	88:8,11,14 89:5,8	113:7
11:17 33:5 34:10	1998 71:19	89:24 92:5 112:20	4.0 5:24 40:16 41:23
84:23 85:9,14,16	1999 3:15	112:24	40 5:23 13:13,14,25
86:3,7,10,18,23,23	1:04 113:15	2500 2:16	14:1 62:23 67:1,3
87:3,21 88:1,9,15	1:27 113:18	27 34:9,11,12,13,15	72:13 117:4
88:16 89:6 91:10,12	1:42 2:18 124:19,23	34:16	41 77:24 78:1,2
91:19 96:14 99:7,17	2	272 23:23	420 110:8,12,15,25
101:13 110:1,9		28 6:17	
111:4,10 112:14,25	2 5:13 11:3,6 12:4	29 23:12,15,19	440 89:14 91:5,9,16
113:21 116:14,17	31:22 62:13,15,20	3	93:24 94:21 96:20 450 94:11,22 96:20
10 5:10 6:8,16 24:10	62:23 63:15 64:6		1 ′ ′
39:13,24 40:6 41:24	74:20 78:21 99:21	3 5:16 15:22 16:1	470 88:21 89:11,24
44:14 104:1,3,4,7	100:1,8 119:12	37:7 116:1,11,17,20	90:10,14 91:3,7,9
104:15	20 6:11 71:22	30 23:19 67:6 72:13	91:17,22 92:4 93:23
	128:17	126:16	94:17,22 96:21
104 6:16	20005 1:22 4:9	31 35:8,11 37:25	111:15
10:45 60:22	20006-1101 3:16	312-701-8641 3:9	4a 106:25 107:1,2,5
10:59 60:25	2001 35:13 37:7,20	32 38:6 64:23 65:6	4th 40:22
10th 10:25 16:11	2002 114:12,14,20	66:10,11,13,18 71:3	5
11 5:13 24:17 33:7	119:25	71:17 74:1 76:1	5 5:21,23 21:7,13
113:23	2003 39:10 72:9	84:21 85:13 110:4	31:24 40:11,15
1100 4:8	76:20 77:21 79:18	326 10:11 12:2	92:15,18 93:7 94:14
111 5:11	79:20 114:3,7	13:19 121:6	94:19 95:2,6,7,7,21
11:46 98:3	119:25 120:25	33 5:18 62:19 64:25	95:23 96:5,6,16
12 6:14 64:10	2011 32:16	65:8 70:10,11 85:23	97:8,21 112:3 116:4
1201 1:21	2014 10:25 11:15	330 11:7,15 12:5	116:5,8
123 5:6	12:17 16:11 38:19	13:21 99:21 100:1	5,808,690 6:5 61:8
1250 1:21	2015 1:17 2:18 7:1,6	34 62:19 63:4 64:6	50 6:5
127 36:25	125:18	65:10,20 67:15 68:4	52 72:21
128 65:9	202-263-3318 3:17	68:22 69:9,20,24	55 72:21
12:37 98:8	202-371-2600 4:10	70:6 72:18 73:8	6
13 21:13,16 22:24	21 25:2,5	75:7 76:23 77:7,8	
23:20	2134835 126:25	85:19	6 6:4 61:2,7
14 23:5,11	127:25 128:25	350 2:16	6,897,871 6:8 61:19
15 1:17 2:18 5:16	214 63:9	36 5:20	60606-4637 3:8
7:1 62:2,4,5 63:9,16	215 87:7	369 14:16,19 100:8	61 6:4,7
64:5 65:22 66:4,24	216 68:19	3d 26:7 30:18	62 88:17
158 102:16	22 25:8 113:23	3d10 40:21	64 74:1
	121:8		
	T.T. 1 T	rol Colutions	

Veritext Legal Solutions 215-241-1000 ~ 610-434-8588 ~ 302-571-0510 ~ 202-803-8830

[66 - architecture] Page 2

7 7 6 :7 61:15,19	91 110:6 92 95:12,16 95 89:10,22 98 6:13	address 57:12 addresses 67:5 adopt 19:13	anybody 46:7 47:7 anymore 71:19
7 6:7 61:15,19	95 89:10,22		_
7 6:7 61:15,19	-		anyplace 102:5
10600		advanced 47:15	anyway 31:20
106:23	9:38 2:17 7:2,5	afternoon 98:12	apart 16:13 17:7
7,015,913 6:11	9th 11:15 16:10	ago 71:22	82:3 83:23 84:2
84:11	a	agree 7:17 45:20	api 39:24
7,376,811 6:14		67:13 68:2,25 69:7	apis 44:13
	a.m. 2:17 7:2,5	73:7,9 74:17,22,25	apologize 24:25
70 17:14	60:22,25 98:3	95:20 101:18	29:11 99:20 103:19
	able 19:20 32:24	105:10,15	112:13
75 29:8	72:4 102:13 118:16	ahead 27:6 81:6	appeal 1:2 2:2 6:16
U	abstract 40:19 87:1	al 56:10	appear 84:3 101:4
1 X 3.3 0.10 40.73	accelerated 30:7	algorithms 26:10	101:21,22
48.1 / 04.74 03.7	accept 105:12	97:11	appearance 7:24
84.6 10	acceptable 46:11	allow 64:8	appearances 3:1 4:1
80 13.18	access 67:22 68:13	alu 52:23 53:9,11	appears 10:21 24:23
84 6:10	accessing 65:11	55:25 56:13,15,17	82:3 100:16 101:9
1 XD 2.14	accumulator 55:8	56:23 57:2,7,11,16	110:12 111:15
87 106:24	accurate 48:23	57:23 58:14 59:3,5	appendix 10:15,19
871 14:14 38:21	86:12 125:11	59:10,11,12,17,19	11:11 14:7
39:1,8 61:20,23	126:19	59:24 60:6 62:19	application 18:23
62:2,5 63:16 64:5	acknowledgment	64:9,11,15,16 65:10	45:2 46:9 47:13
65.22.66.4.25.78.5	128:3	66:20 75:10 77:11	52:16 55:7,10,11
85:2,10,14 86:4,10	acronym 100:16	77:11 124:1,4	114:2
86.19 23 23 90.6	101:19 102:6	alus 66:7 77:9	applications 17:18
91.19 92.10 11 12	action 7:21 125:13	amount 71:3 75:25	32:24 45:8,9,19
92:15,16,22 94:7,14	activate 58:12 59:10	80:9	46:11 47:16
94.24 95.1 1 20 23	117:20	analysis 81:18 82:1	apply 19:20,23,24
96:7,14 97:5,15,23	activates 58:14,14	82:11,20	applying 30:23
101.13 103.10	58:25	angeles 1:16 2:16	appreciate 24:8
106.22 24 25 110.1	actual 56:4	7:1,12	103:1,1
111:25 112:14	add 11:18 12:6 16:6	anido 31:1	approach 46:5,15
114:2	16:14 23:22 41:17	answer 9:9,17,18	46:17 47:2,10
8714 2:20 125:22	57:16 106:11	27:2 42:2,16 50:4	appropriate 126:7
88 21.14	109:23	58:22 59:1 60:3,7	arbiter 86:24 87:3,6
0	added 19:11 32:21	63:19 64:7 67:25	106:14,19 107:7,9
	adding 71:16 114:23	75:9 76:10 89:21,23	107:15,18,23 108:4
	addition 101:21	91:2 101:3 103:2,2	108:14,24 109:12
	additional 16:7,9	109:16 122:9,12	109:20 110:9,16,21
115:16,21 116:17	30:1 57:7 97:7	answered 94:4	110:24,25 111:3,7,9
117:5 118:3	106:2,8,11	111:17	architect 65:14
	additions 40:25	answers 128:7	architecture 5:25
89:3 112:17 113:1	additive 114:14,16	anticipates 85:2	20:18,21,24 21:4
		anticipates 03.2	26:5,14,20 29:22
	Veritext Leo		20.3,17,20 23.22

Veritext Legal Solutions 215-241-1000 ~ 610-434-8588 ~ 302-571-0510 ~ 202-803-8830

[architecture - carried] Page 3

38:23 39:11,25	128:10	basis 17:21,22 35:16	bug 51:19 52:3
40:16,24 41:6,25	attention 104:13	battery 45:18,21	burden 47:17
42:19 43:7,15,22	attorney 3:6,14 7:25	beginning 2:17 7:24	bus 65:13
44:1 50:22 51:13	125:14 126:16	98:6	bytes 65:9
62:15,16 78:8,10,13	attorneys 4:7	behalf 2:15 8:2,6	c
79:5 80:3 99:11	audio 7:16	behest 59:4	
100:21 123:17	author 28:17,18,21	believe 78:3 99:5	c 83:8
architectures 20:13	29:11	best 12:25 42:21,25	c100 31:15
28:1 44:3 99:9	automatic 31:11	better 51:20	c47 24:23
area 57:22	available 18:9,11	big 56:13 70:5	c76 25:2
arithmetic 56:18	34:23 64:16 97:14	bigger 74:25	c78 30:7,12,14,15
64:25 65:8 105:4	97:17,22 106:19	bit 26:22 58:25	c81 25:5 31:2
array 33:1,2 62:22	avatars 31:12,13	64:24 65:7 74:1,1	c83 30:20
63:1 67:2,6	avenue 2:16 4:8	74:20	c85 31:1
art 48:10 75:17 78:4	aware 38:20,25 39:6	bits 74:3	c87 25:8
78:25 80:1 81:14	39:13,23 43:5 48:2	blending 87:18	c89 25:10
82:9,14,25 83:5,11	49:7,19 50:9 103:16	89:17	c90 31:8,9
114:7,19	119:22 120:9	block 54:12 62:8	c91 31:9
article 5:20,23 42:10	awkward 122:23	63:5,20 64:5 65:21	c95 25:12
articles 21:14 22:21	b	91:25 95:12,16	c98 31:13,16
asic 47:2,20 48:21		119:2,6	calculation 89:17
51:16	b 83:3	blocks 58:15	calculations 37:13
asked 42:24 44:21	back 27:9 29:4	board 1:2 2:2 6:17	calibration 31:9
55:24 94:4 103:11	48:21 60:24 61:5	103:16 106:6 122:6	california 1:16 2:17
103:12,18,19	98:1,7,11 109:25	board's 103:9,22	7:1,12 17:13 18:13
111:14 123:25	113:17,21 121:3	105:11,13	125:2
asking 39:19 44:23	background 33:9,11	boarding 97:11	call 13:7 59:11,17
45:10,14 50:7,8	33:19	book 22:24 23:2,2	59:19 60:14 107:23
56:6 86:15 118:2,24	bagherzadeh 1:15	border 90:11	called 22:9 66:19
aspects 101:1	2:14 5:3,11,14,17	borders 90:18,19	calls 42:13
116:12	7:7 8:13 61:5	bottom 33:12	camera 30:19 31:4,9
associated 35:23	124:20 126:2 127:2	104:14	capabilities 32:22
assume 9:10 23:6	128:2	bound 13:11	capability 32:23
101:25 114:4	bank 65:1,9	box 66:19 88:17	47:14
assuming 92:14	based 18:19 31:6	branches 30:22	capable 65:10 68:9
assure 120:18	39:6,22 42:2,10	break 9:13,18 60:19	104:25 105:3,18
ati 1:8 2:8 7:14 8:3	43:21 58:11,23	82:8 97:24 98:13	capacity 17:5
32:3,6 126:1 127:1	92:21 93:9 94:14	breaks 9:12	caption 7:13
128:1	96:16 99:11 104:22	bridges 104:15	car 46:23 60:1,2,5,9
ati's 120:9	121:21 basic 106:10	bring 76:12	60:13,15,15
atlantic 1:21		broad 45:1,10	card 38:13
attached 10:7 11:4	basically 19:16	brown 3:4,12 7:12	carefully 126:5
15:23 16:2 36:8	46:19 47:2 58:24	8:5,7	carried 115:23
40:12 61:3,16 84:7	72:21 86:23 102:15	buffer 86:22 110:8	116:19 117:7
98:20 104:2 126:13	107:21,22	110:12,15	
	X7 '4 4 T	·	

Veritext Legal Solutions 215-241-1000 ~ 610-434-8588 ~ 302-571-0510 ~ 202-803-8830

[carving - content] Page 4

carving 30:8 31:14 case 7:13,14 13:23 circles 27:12,17,19 circuit 56:4,14 87:3 15:2,4,10,18 28:24 29:20 32:6 51:19 111:3,8 circular 26:4 74:21 87:5 101:16 103:20 108:21 13:4 119:12 121:6 cases 10:2,3 32:3,5 32:12 cell 45:13 cellular 7:19 central 118:13 certain 12:23 14:10 39:12 42:20 85:2 112:22 certified 2:19 125:1 certify 125:3,12 128:5 change 16:24 17:1 51:21 71:7 97:25
15:2,4,10,18 28:24 29:20 32:6 51:19 111:3,8 circular 26:4 citation 33:20 combining 102:9 comference 23:5, 113:4 119:12 121:6 cases 10:2,3 32:3,5 cellular 7:19 cellular 7:19 central 118:13 certain 12:23 14:10 39:12 42:20 85:2 112:21 certified 2:19 125:1 cetained 63:4,16 change 16:24 17:1 51:21 71:17 97:25 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
29:20 32:6 51:19 111:3,8 combined 13:14 119:11 55:2 56:1 61:14 circular 26:4 combining 102:9 conditional 30:2 74:21 87:5 101:16 citation 33:20 come 12:24 29:15,17 conference 23:5, 103:20 108:21 120:23 34:1,18 54:22 91:25 123:15 cases 10:2,3 32:3,5 civilian 45:9 63:16 64:5,5 65:21 comes 12:3 46:10 configurable 22: cell 45:13 63:16 64:5,5 65:21 coming 22:19 90:15 configuration 66: configuration 66: cellular 7:19 66:3,24 82:10 85:16 common 38:11 41:6 112:13 confused 50:6 certain 12:23 14:10 85:20 86:3,7,10,18 common 38:11 41:6 112:13 confused 50:6 39:12 42:20 85:2 88:1,9,15,16 89:6 communication 19:7 81:7 99:22 100:6 12:22 91:10,12,19 96:14 19:7 companies 28:13 consider 26:11 4 certified 2:19 125:1 101:13,17,20 110:9 company 18:20 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 </td
55:2 56:1 61:14 circular 26:4 combining 102:9 conditional 30:2 74:21 87:5 101:16 citation 33:20 come 12:24 29:15,17 conference 23:5, 12:34:1,18 54:22 91:25 conference 23:5, 12:31:5 113:4 119:12 121:6 120:23 97:25 101:5 conference 23:5, 12:31:5 cases 10:2,3 32:3,5 civilian 45:9 comes 12:3 46:10 configurable 22: configuration 66: configuration
74:21 87:5 101:16 citation 33:20 come 12:24 29:15,17 conference 23:5, 103:20 108:21 120:23 34:1,18 54:22 91:25 123:15 13:4 119:12 121:6 120:23 97:25 101:5 conference 23:5, cases 10:2,3 32:3,5 civilian 45:9 comes 12:3 46:10 configurable 22: cell 45:13 63:16 64:5,5 65:21 coming 22:19 90:15 configuration 66:3,24 82:10 85:16 cellular 7:19 66:3,24 82:10 85:16 commission 128:19 confused 50:6 certain 12:23 14:10 86:23,23 87:3,20 102:17 confusing 60:4 39:12 42:20 85:2 88:1,9,15,16 89:6 communication 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 84:14,17 99:3,24 127:5
103:20 108:21 cite 102:5 103:4 34:1,18 54:22 91:25 123:15 113:4 119:12 121:6 120:23 97:25 101:5 conferences 120: cases 10:2,3 32:3,5 civilian 45:9 comes 12:3 46:10 configurable 22: 32:12 claim 62:2,4,5 63:9 47:19 configuration 66: cell 45:13 63:16 64:5,5 65:21 coming 22:19 90:15 configuration 66: cellular 7:19 66:3,24 82:10 85:16 commission 128:19 confused 50:6 central 118:13 85:20 86:3,7,10,18 common 38:11 41:6 112:13 certain 12:23 14:10 86:23,23 87:3,20 102:17 confusing 60:4 39:12 42:20 85:2 88:1,9,15,16 89:6 19:7 communication 81:7 99:22 100:6 112:22 91:10,12,19 96:14 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 company 18:20 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compare 57:17 considered 53:9,17 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 compon
113:4 119:12 121:6 120:23 97:25 101:5 conferences 120:23 ases 10:2,3 32:3,5 civilian 45:9 comes 12:3 46:10 configurable 22: 22: cell 45:13 63:16 64:5,5 65:21 coming 22:19 90:15 configuration 66: configuration 66: cellular 7:19 66:3,24 82:10 85:16 commission 128:19 confused 50:6 10:13:13 certain 12:23 14:10 86:23,23 87:3,20 102:17 confusing 60:4 consider 26:11 4 39:12 42:20 85:2 88:1,9,15,16 89:6 communication 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compare 57:17 considered 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
cases 10:2,3 32:3,5 civilian 45:9 comes 12:3 46:10 configurable 22: configuration 66:22:4,5 63:9 commission 12:3 46:10 configuration 66:32:4 46:10 configuration 66:32:10:5 configuration 66:32:10:5 confused 50:6 confused 50:6 confused 50:6 112:13 confusing 60:4 consider 26:11 4 20:13 20:14 1 20:14 1 20:17 companies 28:13 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:14 1 20:1
32:12 claim 62:2,4,5 63:9 47:19 configuration 66 cell 45:13 63:16 64:5,5 65:21 coming 22:19 90:15 configuration 66 cellular 7:19 66:3,24 82:10 85:16 commission 128:19 confused 50:6 certain 12:23 14:10 86:23,23 87:3,20 102:17 confusing 60:4 39:12 42:20 85:2 88:1,9,15,16 89:6 communication 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
cell 45:13 63:16 64:5,5 65:21 coming 22:19 90:15 confirm 100:5 cellular 7:19 66:3,24 82:10 85:16 commission 128:19 confused 50:6 central 118:13 85:20 86:3,7,10,18 common 38:11 41:6 112:13 certain 12:23 14:10 86:23,23 87:3,20 102:17 confusing 60:4 39:12 42:20 85:2 88:1,9,15,16 89:6 communication 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 consideration 80 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
central 118:13 85:20 86:3,7,10,18 common 38:11 41:6 112:13 39:12 42:20 85:2 88:1,9,15,16 89:6 102:17 confusing 60:4 112:22 91:10,12,19 96:14 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 consideration 80:20 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
certain 12:23 14:10 86:23,23 87:3,20 102:17 confusing 60:4 39:12 42:20 85:2 88:1,9,15,16 89:6 communication 19:7 81:7 99:22 100:6 112:22 91:10,12,19 96:14 19:7 81:7 99:22 100:6 102:17 companies 28:13 107:14 114:10 107:14 114:10 128:5 claimed 63:4,16 128:5 claimed 65:21 66:3,24 86:3 51:21 71:17 97:25 86:18 88:14,21 complicated 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
39:12 42:20 85:2 88:1,9,15,16 89:6 communication consider 26:11 4 112:22 91:10,12,19 96:14 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 consider 26:11 4 128:5 claimed 63:4,16 compare 57:17 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
112:22 91:10,12,19 96:14 19:7 81:7 99:22 100:6 certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 consideration 80 128:5 claimed 63:4,16 compare 57:17 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
certified 2:19 125:1 101:13,17,20 110:9 companies 28:13 107:14 114:10 certify 125:3,12 111:4,10 112:14,24 company 18:20 consideration 80 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
certify 125:3,12 111:4,10 112:14,24 company 18:20 consideration 80 128:5 claimed 63:4,16 compare 57:17 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
128:5 claimed 63:4,16 compare 57:17 considered 14:24 change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
change 16:24 17:1 65:21 66:3,24 86:3 compared 53:9,17 35:4 61:11,23 75 51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
51:21 71:17 97:25 86:18 88:14,21 complicated 63:19 81:10 82:20 83:2 127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
127:5 89:25 91:10,19 92:5 component 112:8 84:14,17 99:3,24
changed 17:3 92:11,12 93:22 94:1 components 57:10 103:9
changes 12:1,10 94:12,23 96:23 58:13 65:25 86:8 considering 51:1
16:21 41:1 51:7
126:12 128:9 111:16,25 112:14 88:13,23 89:4,7 consistent 37:23
chapters 22:24 23:2 claims 83:5 85:2 90:9,20 91:18,24 64:11 68:17 72:4
charge 22:15 48:16 101:11,12,22 97:12 106:10 92:24 93:10 94:8
48:17 class 32:16 57:1 111:18 112:5,19,23 95:12 100:21
chart 67:8 85:20 clear 9:1,5 44:7 comprises 64:24 106:13 114:22
86:11 116:11 58:16,20 59:3,5 65:7 consolidate 27:1
charts 111:10
cheaper 46:5 73:1,14 76:21 85:1 computation 30:24 construction 105
check 33:22 100:17 86:21 92:2,3 118:11 33:1 45:17,22 51:17 105:13 106:8
105:9 115:12 clearly 67:18 51:25 68:8 69:15 construe 104:18
checked 120:3
checkmark 24:12 close 77:11,19 computations 99:13 construed 103:1
26:23 closer 55:25 75:10 computer 30:10,16 consultant 5:16
chicago 3:8
chip 20:12,17,21 code 29:22 78:7,19 consumption 47:
71:23 77:7,8 cofounder 19:6 computing 22:6 47:19 51:6,11
chips 52:2 coined 95:2 concept 26:13 51:23 contacted 12:13,
cihlar 4:15 7:8 column 36:25 72:21 concern 75:24 76:8 13:2,4,9
circle 27:8 63:14 99:7,17 115:16,21 concerned 83:19 content 81:13 82
86:2 116:5,8,17 117:4

[context - depends] Page 5

context 44:24 52:9	correction 70:23	curriculum 5:16	declar 44:8
106:16 109:4	corrections 126:5,8	customers 19:13	declaration 5:10,13
continue 20:14,20	128:9	cv 16:2,6,16,19 17:7	10:11,24 11:7,14
61:1 64:20 79:23	correctly 99:19	17:10,12,19,24 18:2	12:2,5,6,11 13:9,10
98:9 113:19	105:8	18:5,9,15,19 19:5	13:15,19 14:6 16:10
continued 4:1 6:1	correspond 87:20	21:7 22:24 23:13,20	24:24 33:5,7 35:9
20:15	87:25 88:8,14 89:5	23:23,23 24:24 25:2	38:7 44:9 49:24
contributed 23:3	89:25 91:9 92:7	25:18 31:24,24	61:24 63:10 70:10
control 58:12 59:10	93:2,22 112:24	32:12 123:3,7	72:15 74:7 77:25
59:13 62:23 67:1,3	corresponding 87:7	cvs 17:4,9,10,12,15	80:12 81:2,8,16,24
86:22 87:4 107:18	90:22 94:15,19	cylinder 46:23	82:5 83:23 84:22
107:20,24,25 108:1	117:21	d	85:6,14,23 87:11
108:2 110:8,12,15	corresponds 63:4,15	d 64:14 83:14	99:20,23 100:1,8
117:20,25,25	64:6 65:20 66:3,24	d.c. 1:22 3:16 4:9	102:4,6 103:3
118:14,14	86:3,18 88:21 91:12	data 19:19 55:18	109:25 112:17
controlled 99:15,18	91:18 93:5 94:1,11	56:12 65:12,24	114:18 120:24
controller 63:1	94:23 96:11,22	66:15 67:14 68:3,21	declarations 10:1,3
conversations 7:19	110:16,21 111:3,16	69:8,19,20,23 70:4	14:24 16:3 61:12
convert 118:18	111:24	70:5,5 71:1,8,25	84:18 99:4
convinced 121:1	cosmetic 17:1	70.5,5 71.1,8,25	decoded 59:4,9,13
core 48:17	cost 45:23,25,25	73:23 74:9,11,23,25	decoder 57:13
correct 8:23,24 9:23	46:4,9,9 47:4,18,19	75:1,7,7,10,13,18	dedicated 71:4
10:13,23 11:1,9,16	counsel 7:10 8:10	76:4,11,14,14,17,22	75:25
14:3 16:4 18:14,20	15:14 45:4 54:8	76:24,25 77:1,3,9	deemed 126:19
23:4,17,18 32:9,11	67:11 79:2 93:12	77:12 78:12 79:5	default 37:14
34:17 38:5,23 39:11	98:14 100:11	87:8 99:9,13,14	define 44:18,20 57:8
48:3,6 49:24,25	108:17 122:5 123:6	100:18,20,25 101:1	57:9 92:9 95:17
50:11,17,20,23 53:3	count 22:13 90:13	101:11 102:20,22	defined 53:13 90:25
53:25 54:24,25	counted 24:5 30:11	122:2 124:8,8,11,12	92:17,22 94:7 96:1
57:17 61:24 62:3,4	counter 57:15 58:23	date 42:20,22	96:10 122:16
62:18,21,23,24,25	couple 8:24 30:21	125:15 126:10	defines 95:1
63:2 67:15 69:21	122:20	128:14	defining 96:6
74:23 75:1 77:4,5	coupled 65:1,9	dated 125:18	definitely 30:9,11
80:21 82:21 83:1,6	course 79:24 109:22	day 128:17	31:4 72:25 83:25
83:12,13,17 84:4,19	court 7:8 9:1 10:7	days 126:16	definition 59:12,23
85:3,4,22,24,25	11:4 15:23 36:8	de 106:12	79:1 92:9,10 93:10
88:10,17 92:18	40:12 61:3,16 84:7	dec 114:17	degree 78:18
100:24 102:7 104:9	98:20 104:2 126:20	december 10:25	delete 11:24 12:8
104:10 106:21	cpu 94:14 95:11	11:15 12:15 16:10	16:18
108:5,15 110:17,18	create 26:6	16:11 38:19	dependent 55:11
112:15 113:1 115:5	creation 31:11	deciding 87:6	depending 52:18
115:11 116:20	credit 28:25 29:6	decision 6:17 103:10	73:22 80:6 97:17
120:1,12,25 122:2	csr 125:22	103:14 104:8,18	106:10 107:24
123:8,9,11 128:6	current 79:19	121:4 122:7	depends 46:2,8 47:6
			47:13 55:7,9 74:2,4

[depends - evolution] Page 6

74:18 80:10 106:12	different 17:10,10	112:11 113:9	eight 23:25 24:5
deponent 128:3	17:17 32:24 53:2,22	dispute 46:7 47:7	25:15
deposing 126:15	53:25,25 54:3,3,7	distinguish 53:6	either 45:15 119:9
deposition 1:15 2:14	54:23 56:4,7,14	document 36:12	electrical 78:19
7:7,11 8:22,25 14:5	60:1 95:4,5 108:17	42:3 61:9 104:11	electronics 1:5 2:5
14:22 15:14,17	108:18,18	documentation	7:13 8:6,8 126:1
126:4,13,17,18	differently 63:22	28:12	127:1 128:1
depositions 13:12	digital 20:7	doing 46:25 68:9	element 62:22 63:1
derivative 89:16	diligence 24:8	69:15 72:9 80:10	64:23 65:6 66:18,19
described 74:1	dimensional 30:8	88:3 100:22 116:24	67:2,3,6 68:13
102:18	31:11,14	119:16 126:9	71:17 77:18
describing 96:7	direct 40:21 104:13	dr 5:10,13 7:7 61:5	elements 62:6 63:16
description 5:9 6:3	directed 23:25	124:20 126:2 127:2	66:10,11,13 71:3
97:9,10 101:17	24:14 27:15	128:2	76:1 86:3,18 118:13
102:1 122:16	direction 125:10	draw 34:24	118:15
descriptions 74:2	directly 27:23 31:18	drive 3:7	embodiment 115:24
81:25	64:25 65:8 73:10	dropping 49:3	117:18 118:6,8
design 19:6 52:2	directx 38:10,22	dsp 19:6 20:10	embodiments
56:4 72:4 73:16	39:10,13,14,24 40:6	dual 53:19,20	118:21
119:18	41:24 43:6 44:14	due 24:8	emphasized 75:24
designed 33:3 53:20	disclo 119:1	duly 8:14	employee 125:14
designer 73:15	disclose 68:20 69:8	duties 19:6	ends 120:19
designers 73:1	disclosed 62:16,17	dutton 4:6 8:4	energy 56:5
designing 80:8	71:19 72:22 76:19	duty 79:15	engagements 17:25
designs 120:4	77:22 115:9 118:2	e	engine 47:21 60:1,2
desktop 45:6,13	119:1 122:1	e 13:3 64:14 127:3	60:5,9,10,11 67:18
destination 89:19	discloses 67:14 68:3	earlier 55:24 100:6	69:14 80:8 99:12
detail 97:19	70:3,16,24 72:16	111:14 112:7 123:3	102:19
details 31:15 63:6	74:22 76:22 85:16	early 12:16 72:9	engineer 20:6 76:20
97:2 106:3	96:14 102:6,19	79:18 114:7	engineering 78:20
detection 27:8	103:5 110:7 121:16	easier 82:7 86:14	entire 33:18
determined 28:20	disclosures 101:20	easy 51:21	entities 54:14
determining 114:5	120:5	editors 21:25	entitled 5:20,23
developing 123:17	discuss 81:17,19,21	efficiency 44:18,20	36:12
123:19	82:4 83:22 97:23	44:21,22,24 45:17	environment 38:14
development 38:13	111:10	45:18,18,19,22,23	equivalent 78:20
device 51:22	discussed 112:3,7	45:25 46:1 48:19,23	errata 126:7,9,12,15
diagram 52:19	121:14	49:1,13,18 50:1	128:11
63:20 119:2,6	discussing 27:21	51:5	evaluated 117:22
difference 53:24	discussion 81:23	efficiency's 45:21	evaluation 118:17
54:19	84:2	49:14	eventually 32:21
differences 53:12	dispatch 91:5,6	efficient 44:15 45:16	89:11
55:20 56:2,9,13	dispatcher 89:14,24	45:24 47:4 51:16,24	evidence 36:20
82:9 83:4	91:16 92:4 93:24	51:25	evolution 34:6 38:4
	94:21 96:20 112:4		43:22

[exactly - form] Page 7

exactly 34:24	expires 128:19	familiar 78:11 79:4	firm 8:2
examination 5:2	explain 72:24 86:14	79:9,25	first 8:14 12:13 13:5
8:11,17 122:25	95:25 96:9 97:1,11	far 51:16	13:8 28:16 29:3,9
examined 8:14	explained 92:2	fast 65:2	29:10,11 35:11,12
example 37:2,6 60:8	108:19	faster 55:21,22	38:9 39:14,24 40:3
70:16,24 79:25	explaining 97:20	103:24	40:8,20 41:15,24
110:22,23 114:11	explicitly 68:20 69:5	features 41:7	42:19 73:3 83:19
excellent 67:9	69:12 70:3 76:21	february 37:7	98:4 115:21,23
excited 67:11	exploded 97:8,9	feed 91:24	116:18 117:17
excuse 95:22 116:6	exploiting 5:23	felt 19:19,22 44:1	118:3 120:11
execut 87:25	40:15	ffp 46:18	fish 32:14
executable 77:9	exponentiation	field 34:2 78:22	fit 70:5
execute 90:10	37:12	79:22 99:6,17	five 23:2 72:25
executed 115:18	expressed 44:9 81:8	100:12	fixed 46:18 48:3
117:21	84:18 99:4 100:7	fields 22:16	49:9,20 50:10
executes 58:10,22	expression 118:17	fifth 28:18	fixes 51:19
64:17	extent 26:20	figure 26:8 34:10,15	floating 74:1 105:3
execution 48:19	external 70:25 71:9	34:18,21 62:13,15	flow 116:11
87:14,25 88:4,7,14	71:25 72:8,17 73:8	62:20,23 63:15 64:6	follow 29:1,8
88:18,20 89:4,7,11	74:23 75:8,13,19	64:10 76:20 85:24	follows 8:15
89:24 90:14,25 91:3	76:22 77:3 124:9	86:1,2,6,16,17	foregoing 125:4,6
91:16,17,21 93:23	extremely 71:17	88:12 90:16 92:13	125:11 128:5
94:17,21 96:21	eye 1:21	92:15,18 93:7,9,11	forgetting 28:4
111:15 112:6,19,23	f	93:21 94:14,19 95:2	form 11:19 13:3
113:3,4	facilities 41:4	95:6,6,7,21,23 96:5	15:3,11 18:21 19:25
executive 17:19	fact 62:1 73:18	96:6,16 97:8,21	24:2,19 25:19 27:16
exhibit 5:10,13,16	75:11 100:15,23	106:25 107:5	27:22 29:13 32:8
5:20,23 6:4,7,10,13	101:21 102:13	110:13 111:16,24	33:21 35:25 38:24
6:16 10:6,10,16,17	103:16 116:17	112:3,21 113:7	39:16 40:2 42:1,13
11:3,6,17 12:4	120:6,9	116:1,4,11,13,17,20	43:1,8,17,24 44:10
15:22 16:1,2 33:5	factor 51:9 80:20	119:12	44:17 47:11 48:11
36:7,11,16 40:11,15	factors 80:17,21,22	file 57:2,12,23 58:14	49:10,22 50:12,19
41:13 61:2,7,15,19	80:23 81:3,8,10,12	94:6,10,22 95:18	50:25 51:14 52:24
84:6,10 85:9,14	81:13,17,24 82:1,4	96:20 112:6 113:2	53:4,10 55:14 56:24
98:17,19,22 99:3,21	82:16,19 83:15,19	filed 38:22 39:1,8	58:2 59:8,18 63:17
100:1,8 104:1,7	83:20,22 84:2	114:3	65:23 66:5 67:16
106:23 110:1	faculty 28:23 29:1,7	financially 7:22	68:5 69:10,25 73:20
113:21	29:18	125:13	75:2 80:5 81:20
exhibits 5:8 6:2	fadi 22:7	find 18:18 30:1	86:5 90:1,7,24
14:23	fail 126:18	47:16 51:7 73:4	91:13 94:25 96:25
exotic 26:9 46:20	fair 9:3,10,11,19,20	84:5	101:24 107:8,16
experience 17:25	84:15,16	finding 26:4	108:6,16 109:1,14
31:22 78:22 123:13	fall 12:16	fine 16:8	112:1 114:21 115:6
expert 34:3 41:23		finish 26:24 93:14	116:21 119:5 120:2
		93:15	120:13 122:3 128:9

[format - huh] Page 8

format 73:22 generate 34:21 gpus 5:20 36:12 hand 36:25 84:18 99:3 100:7 44:8 81:7 generates 109:17,20 grab 103:25 36:10 40:14 84:9 forth 125:5 41:1 115:25 116:13 gradual 114:25 104:6 foundation 17:11 17:8 grand 2:16 handled 31:4 49:11 120:14,19 116:12,18,19,22 47:1 handled 31:4 122:4 117:6,14 118:4,4 grand 4:15 7:8 handled 31:4 four 80:16 81:13,14 119:3 grant 4:5 7:8 handled 88:18 handled 31:4 handled 31:4 handled 31:4 four 80:16 81:13,14 119:3 grant 4:5 7:8 happen 118:22,25 fox 4:4 8:2 give 13:11 17:22,22 78:7 86:24,25 101:1 happening 114:24 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hardware 30:7
84:18 99:3 100:7 generation 40:22 grab 103:25 36:10 40:14 84:9 forth 125:5 41:1 115:25 116:13 gradual 114:25 104:6 foundation 17:11 117:8 grand 2:16 handheld 31:4 49:11 120:14,19 116:12,18,19,22 47:1 handling 119:15 122:4 117:6,14 118:4,4 grant 4:15 7:8 happen 118:22,25 four 80:16 81:13,14 119:3 grants 17:11 119:14 82:19 getting 48:18 graphic 45:4,5,8 happen 30:5 fourth 28:18 80:20 give 13:11 17:22,22 78:7 86:24,25 101:1 happening 114:24 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 119:8,11 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:13
foundation 17:11 117:8 grand 2:16 handheld 31:4 49:11 120:14,19 116:12,18,19,22 47:1 handled 88:18 122:4 117:6,14 118:4,4 grant 4:15 7:8 happen 118:22,25 four 80:16 81:13,14 119:3 grants 17:11 119:14 82:19 getting 48:18 graphic 45:4,5,8 happen 118:22,25 fourth 28:18 80:20 give 13:11 17:22,22 78:7 86:24,25 101:1 happening 114:24 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hardware 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
36:21,23 40:4 41:14 geometry 71:10 grandma's 46:4,14 handled 88:18 49:11 120:14,19 116:12,18,19,22 47:1 handling 119:15 122:4 117:6,14 118:4,4 grant 4:15 7:8 happen 118:22,25 four 80:16 81:13,14 119:3 grants 17:11 119:14 82:19 getting 48:18 graphic 45:4,5,8 happen 118:22,25 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 happening 114:24 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hardware 30:7 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
49:11 120:14,19 116:12,18,19,22 47:1 handling 119:15 122:4 117:6,14 118:4,4 grant 4:15 7:8 happen 118:22,25 four 80:16 81:13,14 119:3 getting 48:18 grants 17:11 119:14 82:19 give 13:11 17:22,22 78:7 86:24,25 101:1 happened 30:5 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 119:8,11 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
122:4 117:6,14 118:4,4 grant 4:15 7:8 happen 118:22,25 four 80:16 81:13,14 119:3 grants 17:11 119:14 82:19 getting 48:18 graphic 45:4,5,8 happened 30:5 fourth 28:18 80:20 give 13:11 17:22,22 78:7 86:24,25 101:1 happening 114:24 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 119:8,11 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
four 80:16 81:13,14 119:3 grants 17:11 119:14 s2:19 getting 48:18 graphic 45:4,5,8 happened 30:5 fourth 28:18 80:20 give 13:11 17:22,22 78:7 86:24,25 101:1 happening 114:24 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 119:8,11 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
82:19 getting 48:18 graphic 45:4,5,8 happened 30:5 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 happening 114:24 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
fourth 28:18 80:20 give 13:11 17:22,22 78:7 86:24,25 101:1 happening 114:24 fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 119:8,11 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
fox 4:4 8:2 28:25 29:20 42:16 graphics 18:20,24 119:8,11 frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
frame 38:17 39:1 42:20 89:21 98:17 19:10,12 24:1,14,16 hard 73:21 77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
77:21 79:16 114:25 given 44:13 97:20 25:3,6,8,10,13,16 hardware 30:7 120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
120:4 119:21 124:20 26:7,14,19 27:13,15 38:11 44:15 99:12
front (1.7.10.00.22 120.7 27.20.21.20.0.20.2 102.10
front 61:7,18 98:22 128:7 27:20,21 28:8 30:2 102:19
function 46:18 59:4 gives 46:1 47:14 30:6,9,10,10,17,20 hardwire 46:5,15
71:10 88:17 115:23
116:19 117:7 giving 58:25 32:6,13,21 33:1 hardwired 51:24
functional 102:21 glad 24:9 34:3,4,7 35:21 heading 33:8 85:3
102:23 glanced 14:8,20 38:13 39:23 40:23 heard 43:11 52:7
functionality 19:11 go 7:17 24:6 27:2,6 41:23 43:22 44:15 56:15,19 106:14
54:20 55:15 65:15 27:6 28:2,15 29:25 44:25 45:3,12 47:8 hearsay 41:14
functionality's 56:7 46:12 47:15 48:14 47:21 48:2 49:8 held 7:11
functions 57:16 51:4 52:1 81:6 82:6 50:17 52:17 67:17 help 29:5 60:6 63
68:15 95:10 82:11 86:1,12 68:7 69:15 78:11 90:9 115:19
further 53:8 71:8 113:13 79:4,17,19,20 80:2 helped 29:23
124:15 125:10,12 goes 26:16 27:8 41:9 80:8 100:22,24 helps 27:11 60:8
future 50:23 78:16 88:23 111:19 101:4,9,14,19 102:7 hereto 10:7 11:4
g going 9:5,10,12 103:6 106:17 15:23 36:8 40:12
13:11 36:19 41:12
gained 78:17 114:15 41:20 42:9 43:3 120:11 123:3,7,13 104:2
gears 33:4 103:8 46:3,13,24 48:21 great 66:21 6/:/ hierarchical 26:1
115·2 60:18 84:14 95:7 grocery 46:24 26:18,20
geforce 37:6 97:20 109:25 124:7 ground 8:24 84:22 high 63:20
general 20:24 38:3 goldstein 4:4 8:2 122:6 history 14:8,19 3.
38·20 39·7 22 43·21 good 7:4 8:19,20 9:4 guess 23:12 42:23 108:12
62.7.63.5.64.4 18:10 48:20 60:19 42:24 43:2,3,25 hold 54:8 /6:1/
65:21.78:12.79:5 67:12.76:3.78:6 92:3.97:25 92:15
82:8 88:3 95:15 98:12 guessing 12:16 hope 52:3
109:4 115:13 google 18:18 guys 92:22 hour 60:19
generalize 55:11 gph 32:14 h hours 13:13,14,18
generally 52:25 gpu 38:3 100:16,20 hammer 19:14 13:25 14:1
93·21 119·22 101:19 102:7 103:6 huh 20:16 28:19
36:17 46:16 62:14

[huh - jzhu] Page 9

83:2,7 116:7 119:24	includes 57:2 60:2	93:5,6,8,12,18,24	ipr 10:11 11:7,15
hundreds 120:4	60:10,11 66:19	94:21 95:8 96:20	12:2,5 99:21
i	92:19 93:11 94:5,6	104:24 105:18	ipr2015-00326 1:7
	94:6 102:20	111:20 112:4,11	2:7 7:15 104:9
idea 29:21,22 42:11	including 88:23	113:8,8	ipr2015-00330 1:7
42:18 43:18	92:17 111:18	instructions 57:19	2:7 7:15
ideas 28:23,24 29:15	inconsistency 76:18	57:25 58:5,9,10,12	iprs 12:14
29:20 42:9,10	inconsistent 77:20	58:17,23 59:6,7,14	irrelevant 36:20
identical 56:8	incorporated 7:14	67:5 89:13 90:15	76:14
identification 10:6	increase 51:12	91:4,6,25 92:21	irvine 18:13
10:10 11:3 15:22	independent 118:1	93:2 94:5 105:1	issue 21:22,24 22:2
16:1 36:7,11 40:11	index 5:1 6:1	111:24 126:3	22:4,5 51:6 71:19
61:2,15 84:6,10	indicate 17:24	interchangeably	82:10 83:5
98:19 104:1,7	indicating 81:3,9	124:5	issued 114:2
identified 31:2	83:15	interconnect 102:22	issues 56:1
88:16 93:9 96:4	indication 119:13	interconnecting	itc 32:14
97:21 112:3	120:17	65:13	
identify 31:5 53:12	indirectly 27:23	interested 7:22	j
54:12 82:15 86:7	31:19	14:11 44:2,3 125:13	j 4:6
89:9	indistinguishable	interesting 28:22	j12 26:12
identifying 27:9,17	54:15	interference 7:20	j14 27:8
96:2 106:1	industry 49:8,20	internal 58:13 76:16	j19 28:1
ieee 22:14	50:9,15,18 54:6,22	124:12	j40 28:3
iii 33:8	120:10	internet 34:23 35:2	j41 28:3
illinois 3:8	information 33:25	35:5	j54 24:7,10,11,13
illustrated 116:1,20	52:13 53:13,14,23	interpolation 68:14	j7 26:4
image 26:5,12,18	54:4,4 55:17 93:7	89:16	j71 24:16
31:6,9 115:25	97:7 117:2 119:7	interpret 44:4 72:7	j78 30:13
116:13 117:8	informed 114:1	91:20	jo3 22:6
118:20	initially 43:14 48:3	intricacies 79:12	job 77:8 79:15,23
imagine 28:5	input 87:8 102:14	introduce 41:24	80:7 88:3
imperative 126:14	105:5 109:4,6,17,20	introduced 35:13	jog 41:22
implement 44:16	inputs 87:4,7 104:23	37:8 41:4	john 3:13 8:7
107:17 122:15	106:20 107:19,21	introduces 40:25	jonathan 4:5 8:1
implementing 29:23	107:22 108:1,20	introduction 36:20	journal 21:8,14,17
implicitly 69:6	109:17	invention 73:11	21:22,24 22:2
important 45:22	inquiry 80:15	96:6 99:7,8,17	journals 21:12,17
49:14 56:1	instance 22:5	100:13	22:9,20,23 120:18
improve 30:22 73:3	instituted 122:6	inventor 29:11	123:6
improved 72:12	institution 103:9,13	inventors 63:21	jtuminar 4:11
include 104:24	104:9 121:3	investors 28:6	judging 41:16
105:17 123:7	instruction 58:11,24	inviting 18:5	101:13
included 57:22 82:2	59:4,9 66:15 88:24	involved 13:24 32:2	jzhu 3:18
106:2	89:14,20,24 91:5,5	32:5	
	91:6,16 92:4,19		
	71.0,10 72.1,17		

[k - memory] Page 10

k	lacks 122:4	25:17 28:16 29:10	major 44:13
k 3:15	large 33:8 76:16	32:12 35:6 36:25	mapping 87:18
	larger 73:19 76:4,11	80:20 123:5	mark 86:15
keep 76:4 114:23	77:13	lists 83:14	marked 10:5,6,9
kessler 4:4 8:2,4	latest 40:6,7 46:21	litigation 31:21	11:3 15:22,25 36:6
key 20:6	launched 37:7	little 18:4 26:22	36:7,10 40:10,11,14
kind 29:4 32:25	law 3:6,14 4:7 8:1	55:21 56:7 58:25	61:2,15 84:6,9
48:10 77:13	leading 123:10	76:8 122:23	98:19 104:1,6
kizhepat 14:9 98:23	leave 76:11	llp 3:4,12	master's 78:18
98:25 99:22 100:7	legal 1:21 7:10	local 73:2,8 76:23	match 96:4
100:13,14,16,23	81:25 105:14	77:14	materials 14:23
101:4,10,18 102:5,6	124:22	logic 56:18 64:25	35:4 99:24,25 103:8
102:9,18,19 103:4,5	letting 93:15	65:8 108:2	matrix 37:11
103:5	level 26:18 46:10	logical 105:4	matter 77:9
knew 114:7,14	63:20 102:15	long 31:9 56:16	mayer 3:4,12 7:12
know 9:7,14 16:25	106:12	look 14:16,18,21	8:5,7
17:1,18 18:1,10	lg 1:5 2:5 7:13 8:6,7	15:1 19:4 21:6 24:7	mayerbrown.com
19:14 22:16 24:3	32:3,5 126:1 127:1	26:1 31:21 33:4	3:10,18
27:11 30:4 35:17	128:1	35:4 36:24 44:2	mean 15:14 16:7,23
39:21 42:6 43:11	life 20:25 108:12	62:5,13 63:6,6	24:24 29:11,21
48:19 52:8 55:4	lighting 37:16	68:18 74:6 77:24	30:13 31:15 41:16
58:20 60:15 73:25	limitation 105:11	85:13 99:6,23,25	45:16,23 49:1 51:1
76:8,9 77:18 80:1	117:19	101:8 103:21,21	52:18 54:2 55:7
83:20 84:5 87:24	limitations 118:22	115:14,16,21 117:4	60:13 68:6 76:9
89:2,7 90:10,19,20	limited 73:1,15 80:9	looked 14:6,7,9,10	79:3,8,21,21 82:6
92:4 93:20 96:18,23	116:24	14:14 15:7 26:19	83:25 91:1 101:6
105:15 109:3,7,23	limiting 117:24	32:18 44:1 101:5,6	108:9 109:23 116:5
112:8 118:25	lindholm 84:11,15	101:12 103:13	meaning 108:8
knowledge 34:2	84:17 85:2,16,24	looking 19:14 67:20	means 22:3 23:7
35:17 38:3,21 39:7 39:22 43:21 77:21	86:1,2,16,16,17,22	72:1 92:16 100:19	29:12 45:17,25
	86:25 89:9 91:15	102:8 118:11,12	47:14 56:17 58:13
78:6,18 80:9 82:13	92:5,7 93:4,21,25	looks 52:20,20	58:17 92:19 104:18
83:9 114:15,24	94:11,20,22 95:21	los 1:16 2:16 7:1,12	105:11,19 108:14
119:21	95:24 96:8,13,19,22	lot 28:8 45:21 47:15	116:18
knowledgeable 34:4 34:6	97:19,22,23 101:14	48:17 55:22 97:19	meant 34:12 38:19
known 51:23 114:11	102:9 110:7,13,19	lots 48:20	63:21
114:19	111:6 112:22 113:7	lower 13:11 45:25	mechanical 59:25
kurdahi 22:7	lindholm's 14:9	lunch 98:5	media 98:4,7 124:21
kurihara 14:10	87:14	m	meet 111:7
	line 116:5,6,10,17	maas 12:20	memories 53:19,20
l	117:4 127:5	machine 125:9	memory 41:22
labeled 85:16	linear 89:15 118:16	mail 13:3	52:23 53:2,8,14,16
lack 36:21 40:4	list 30:11	main 41:4	53:22 54:1,14,24
41:13 49:10 120:13	listed 14:23 21:13	ALAMARI TI.T	55:13,22 56:3,6,11
	22:21 23:2,13,20,23		58:11,24 62:19 63:4
	1	I .	I.

[memory - oh] Page 11

64:6,15 65:10,12,20	modify 51:21 69:19	names 12:24 29:3	nw 1:21 4:8
67:15 68:3,13,22	77:3	54:3,3,23	0
69:9,20,23 70:5,25	morning 7:4 8:19,20	national 17:11	o 64:14
71:4,9,16,23,25,25	103:11	necessary 71:2,24	oath 9:22 125:7
72:3,8,12,17,18	morpho 19:2,3,4	126:5	object 11:19 15:3,11
73:2,4,8,8 74:23	28:5 30:6 32:20	need 9:13 42:8,15	15:11 18:21 19:25
75:7,8,13,19,25	123:16	75:9 77:18 79:14	24:2,19 25:19 27:16
76:16,23,23 77:3,7	morphosys 19:1	107:18,18 118:10	27:22 29:13 32:8
77:8,12,13,14 124:9	29:21 32:22 33:2	121:1 122:9	33:21 35:25 36:19
124:13	mosey 77:13	needs 65:25	38:24 39:16 40:2
mention 28:11	motion 30:18	neighbors 65:13	41:12 42:1,13 43:1
44:11	move 51:12	neither 38:22 39:10	43:8,17,24 44:10,17
mentioned 17:16	moved 48:5 49:8,19	101:18 125:12	47:11 48:11 49:10
27:19 30:12 31:18	49:20 50:9,10	never 18:19 32:6	49:22 50:12,19,25
46:13 56:12 81:11	moving 47:9 50:16	67:14 68:2	51:14 52:24 53:4,10
86:7 102:1	multi 52:14 53:18	nevertheless 121:10	55:14 56:24 58:2
merit 17:12	55:3,6	121:14	59:8,18 63:17 65:23
mesh 31:6	multifunction 64:24	new 4:8 40:24 52:2	66:5 67:16 68:5
metric 30:18	65:7	newhart 2:19 7:9	69:10,25 73:20 75:2
microphones 7:18	multimedia 19:7,11	125:21	80:5 81:20 86:5
microprocessor	multiple 30:19	nicely 19:18	90:1,7,24 91:13
57:4,8,9 59:11,24	66:15 117:23,24	non 81:4	94:25 96:25 101:24
124:4	118:13	nonobviousness	107:8,16 108:6,6,16
microprocessors	multiplexers 102:10	81:9 83:16	108:16 109:1,14,14
124:1	102:12	nonvolatile 54:18	112:1 114:21 115:6
mid 1:21 114:7	multiplexing 102:17	normally 66:14	116:21 119:5 120:2
middle 29:4	multiplication 37:12	nos 1:7 2:7	120:13 122:3,4
midway 37:1	mux 107:4,7,10,15	notary 7:20 128:21	objection 94:4 109:1
milberg 32:16,16	107:19 108:20	note 7:16	123:10
military 45:9	109:4	noted 124:23 126:12	objections 7:23
mind 12:3,24 38:1	n	128:10	objective 80:21,23
44:22 45:10,12	n 64:14	notice 28:15	81:3,8,17,24 82:4
55:12 73:22	n.w. 3:15	noticing 7:25	83:15,20,22 84:2
minimum 106:4	nader 1:15 2:14 5:3	november 114:3	objects 27:10
minor 71:17	5:10,13,17 7:7 8:13	nsf 18:9	obvious 62:3 69:19
minutes 64:8	124:20 126:2 127:2	number 5:9 6:3,5,8	75:9,12,16 77:2,6
mis 49:5	128:2	6:11,14 33:8 61:8	118:10
missed 106:7	nadia 2:19 7:9	61:19 84:11 98:23	obviousness 80:15
misspelling 16:25	125:21	99:21 100:1 124:21	81:4,4,9 82:1,20
misspoke 49:6 100:6	nail 19:15	numbered 10:22	83:16
mistake 59:16	name 7:8 29:9	numbers 7:15	occur 117:14
mistakenly 59:11	125:16	nvidia 32:16 37:6	office 1:1 2:1 32:3
model 5:24 39:15	named 28:17,18	43:12,14,25 46:6	oh 13:11 16:20
40:16 41:23 44:14	29:10,11	95:4	33:15 60:20 64:21
51:7 96:9			

[oh - people] Page 12

85:8 121:23	ones 17:17 22:12	output 89:17 107:23	77:24 78:2 80:12,13
okay 8:24 9:5,9,12	30:1 40:7	109:5,6,18,21	82:4,21 83:23 84:3
9:16,21 10:1,4,24	open 94:17	outputs 108:20	87:10,13 88:5 89:3
11:2,10,14,17 12:4	opengl 14:10 38:10	109:17	89:10,22 99:19
12:13,19,22 13:2,8	38:22 39:10 43:6	outside 76:4,12	102:16 110:6
14:2,4,12,18,21	44:13	77:12 123:14	112:17 113:1,23
15:1,9,21 16:13	opening 22:3,8	overview 22:1	121:8,9
17:4 18:3,7 19:4,22	operands 64:11,14	owner 1:9 2:9,15 4:3	parallel 19:16,19
20:7,23 21:6,21	64:16 65:17 95:10	7:11 8:3 121:15	122:15
22:20,23 23:5,11,15	operating 66:10,14	owner's 15:1,9	part 38:12 68:16
23:19,25 24:8,11,11	operation 92:1	р	87:2 91:21 92:21
24:13,16,23 27:4,6	117:12 119:13	p 64:14	93:19 95:17
27:19,25 28:15	operations 67:19	p.m. 2:18 98:8	particular 22:4 28:9
29:19,24 31:21 32:2	68:7 74:12 77:19	113:15,18 124:19	88:17 115:24 118:6
32:20 33:6,11,18	87:16,17 89:15	124:23	particularly 99:10
34:3 35:1,8 37:24	99:13 100:22	page 5:9 6:3 10:15	parties 7:17 125:14
38:20 41:11 43:5,11	104:19,20,21,22	10:18,22 11:11	partition 119:14
45:7,12,17 48:5	105:2,2,5,20,21	17:12,19 21:7,13,13	parts 119:16
49:18 50:8,15,22	115:4,5,10,10	21:16 22:24 23:5,11	party 7:21
51:9,15 52:6,14,22	117:14 118:4,5,5	23:12,15,19,19	patch 91:5
56:9,15,19,22,25	119:3,4,17 121:17	24:10,17,24 25:2,5	patent 1:1,2,9 2:1,2
59:2,5,16 60:16,17	121:18,25 122:1	25:8,12 31:22,24	2:9,15 4:3 6:4,7,10
61:18 62:1,5,11,12	opine 39:21 72:6,16	33:7,12,12,16 34:9	6:13,16 7:11 8:3
63:2,13 64:19,21	75:11,14 77:2	34:11,13 36:15,16	14:8,8,9,14,16,19
66:2,9,12,17,21,23	110:20	36:24 67:9 70:10	15:1,9 32:3 38:22
67:11,11,24 68:11	opined 108:4,14,24	84:21 85:13,19,23	39:8 61:8,8,11,19
68:18 69:4,13,18,22	109:12 111:13	95:13 104:14,15,15	61:20,23 62:3,6
70:13,20 71:15,21	opinion 62:2 63:4	106:24 110:2,4	63:16 64:5 66:4,25
73:12,18 74:13,25	65:20 69:18 85:1	113:23 121:8 127:5	78:5 84:11,15,17
75:6 76:3 77:6 79:3	86:21 89:2 90:20	pages 5:11,14,18,21	85:3,10,15 86:4,10
81:1 82:19,23 83:18	96:13,18,23,24	5:25 6:5,8,11,14,17	86:19 90:6 91:19
83:21 84:21 85:5,22	107:15 111:15,23	17:14 128:6	95:23 96:7,15 98:23
91:9 92:16,23 94:16	124:8,11	paper 24:14,17	100:8 101:23
94:20 95:20 96:6,17	opinions 44:8 81:7	27:14,21 29:12	103:10 106:23,25
97:6 98:16 99:6	84:18 99:4 100:7	papers 16:8,9,13	108:17 110:1
100:2,10,15,17	opposed 80:24	20:17 21:3,7 22:18	111:25 112:14
101:11 103:1,8,23	opt 46:4	23:6,13,22,23 25:15	114:3 121:15
104:13 105:10	order 28:20	28:16 29:25 30:21	patents 13:24 15:8
106:14,19,22 110:5	ordinary 75:17 78:4	44:2 123:14,15	120:5
111:2,12,14 112:17	78:25 80:1 82:13	paragraph 34:12,15	paths 99:14,18
113:12 114:17	83:10 114:6,11,19	34:16 35:8,11 37:25	peer 22:9,12
115:2,3,17 116:4,15	original 49:25	38:6 41:16 63:9	pending 9:17 121:22
117:1,4 118:24	126:15	68:19 70:10,11,12	people 26:19 27:10
119:21 122:8,19,22	outcome 7:22	72:15 74:6 75:15	48:20 52:2 59:11

[percent - processors] Page 13

percent 29:8 72:13	104:20,21 105:2,20	114:24 118:8	problems 19:19
perfectly 33:2	115:4,10 116:25	pointing 91:11 92:7	51:18
perform 37:15 68:7	117:12,14 118:5	93:22 94:1 96:22	proceeding 7:23
89:15 92:1	117:12,14 116.5	110:20	108:13,24 109:12
performance 51:10	123:17,20 124:8,11	ponder 41:5	proceedings 125:4,6
51:11,12 56:13	pixels 31:15 68:14	port 53:19,20	125:8
performing 37:11	73:24 74:9,19 97:16	ported 52:14 53:16	process 95:15
67:19 104:19,20	118:18	53:18 55:3,6,9	processed 89:18
105:1,19 121:17	place 7:17 125:5	portions 14:10	processes 89:12
122:1	placed 125:7	118:19 119:14	91:3
performs 87:15	please 7:16,23 27:6	posita 72:1,6,9	processing 18:20,24
99:12 115:4,9	58:8 61:1 98:9	77:17,21	19:12,21,23,24 24:1
period 55:17 114:5	113:19 126:4,9	position 43:19 52:3	26:12,16,17,19
periodic 9:12	plurality 87:4	111:9 122:14	27:21 28:9,9 32:25
persepolis 31:3	102:21 104:23	possible 28:7 44:4	33:16 40:23 44:25
person 78:3,10,17	plus 22:7 104:23	119:18 120:6	45:3,4,5,8,12 48:2
78:25 79:4 80:1	pluta 3:5 5:6 8:5,5	power 45:18,21 47:5	49:8 50:18 52:10,17
82:13 83:9 114:10	11:19 15:3,11 18:21	47:18,19 49:13 51:6	62:22 63:1 64:23
114:18	19:25 24:2,19 25:19	51:6,11,24 56:5	65:6 66:6,10,11,13
pertinent 82:14	27:16,22 29:13 32:8	pp 35:13	66:18,19 67:2,6,18
83:10	33:21 35:25 36:19	practical 78:21	68:12,16,21,22 71:1
petitioner 1:6 2:6	38:24 39:16 40:2,4	practice 77:16,17	71:2,16 72:5 75:20
3:3 121:15	41:12 42:1,13 43:1	preamble 101:13	75:21 76:1 77:18
ph.d. 1:15 2:15 4:5	43:8,17,24 44:10,17	preparation 15:7	78:7,12 79:5 86:24
5:3,17 8:13	47:11 48:11 49:10	30:23	86:25 99:10 100:18
phone 13:7 45:13	49:22 50:12,19,25	prepare 14:4	100:20,25 101:1,2
46:4,14 48:16,18	51:14 52:24 53:4,10	preparing 13:10	101:12,15 102:20
phrase 100:24	54:8,10 55:14 56:24	14:21 61:12,24	106:17 113:2
pick 7:18	58:2 59:8,18 60:18	present 99:8	116:12,18,20,23
piece 52:23 80:8	63:17 65:23 66:5	presentations 17:22	117:6,15,24 118:13
pieces 96:4	67:16 68:5 69:10,25	primarily 67:3	118:14 119:22
pipeline 35:12,19	73:20 75:2 80:5	primitive 26:6 27:9	123:3,8,14,18,21,22
37:20 40:24 41:2	81:20 85:6,11 86:5	primitive's 117:22	processor 19:16
46:18 48:3,6,8,10	90:1,4,7,24 91:13	primitives 26:7,8	20:23 21:3 48:17
49:9,20,21 50:10,11	94:4,25 96:25	27:9 117:16,20,23	55:25 56:19,20,22
50:16,17 87:15,19	101:24 107:8,16	118:18,19 119:15	57:2,3,4,6,19,22,22
87:25 88:7 89:4,8	108:6,16 109:1,14	prior 41:1 81:14	57:24 58:4,9,10,17
112:20,23	112:1 114:21 115:6	82:9,25 83:4 120:21	58:22 59:17,20 60:5
pixel 19:21,23,24	116:21 119:5 120:2	125:7	62:8 66:3 72:17
26:17 28:9 30:8	120:13,16 121:22	private 7:19	76:4,12 88:2 100:24
32:25 67:22 69:20	122:3,20,23 123:1	probably 26:21	103:6 105:3 120:10
69:23 70:4 73:17,18	123:12,23 124:15	46:21 47:5 72:12	120:11 123:20
74:3,25 75:7,12,18	point 26:23 28:22	problem 48:24	processors 20:8
75:20,21 76:14,24	36:21 48:18 54:5	72:14 73:4,6	47:9 102:14 119:23
77:3,10,20 87:8,17	55:17 74:1 105:4,14		
	Veritext Lea	1014	

[product - relates] Page 14

product 35:22,23	purpose 33:3 62:8	rasterizations 68:14	124:18 125:8
36:1 37:19,22,25	63:5 64:4 65:21	read 35:18 37:5	recorded 7:6
products 37:22	95:16	40:19 54:13 64:1	recording 7:16
120:20	purposes 10:10 16:1	69:11 70:15,21 72:7	recovering 30:18
professional 17:5	36:11 84:10 104:7	78:16 81:14 86:20	31:3
professor 18:12	104:17	97:15 99:19 100:12	refer 21:20 61:20
program 37:14 48:9	put 24:12 26:22	104:16 105:8,25	63:23,23 70:7,9
57:15 58:23	29:3 46:13 57:21	110:8 113:25	84:14 86:11 101:11
programable 35:12	72:3 84:1 90:11,18	116:17 121:13	115:13
programmable 37:8	90:19 92:14,20	126:4 128:5	refereed 23:9
37:20 40:22 41:7	pyramid 26:5,13	reading 54:16 55:16	reference 98:24
48:1,5,20 49:4,21	28:1	95:14,15 105:22	99:22 115:14 124:7
50:11,16 51:17,18	q	122:5	references 14:7
52:1	qualify 48:25 73:13	reads 21:8,17 23:16	34:22 108:18
programmer 37:10	quality 48.23 73.13	35:11 38:9 70:14	referring 35:19
programs 78:9	ques 124:15	71:7 78:2 80:13	37:19 38:18 72:21
promotions 17:12	question 9:7,17,18	84:22 87:13 99:7	79:9,16 85:7,9
22:13	15:14 18:10 39:5,19	110:6 115:22	92:14 93:4,7 95:6,6
propounded 128:8	41:20 44:21 45:1,11	116:10	95:21,23 112:8,21
prosecution 14:18	49:18,25 50:2,4,4,6	really 19:18 22:8,13	refers 108:14
proud 19:18	50:8 55:24 58:4,7	29:14 33:1 46:2,8	regarding 102:9,11
proven 47:4	60:4 63:19 64:7	46:22,22 47:6 55:9	104:8 122:12 123:2
provide 29:21 37:9	67:24,25 73:5 76:10	55:10 76:14 80:10	region 1:21
43:6 44:14 64:9	82:9,17,18 83:18	reason 9:24 126:6	register 52:7,8,9,12
65:15 77:8 120:11	86:8 88:13 89:1,23	127:7,9,11,13,15,17	52:14,22 53:3,9,13
provided 38:23	91:20 103:2,3	127:19,21,23	53:17,21,25 54:14
39:11	105:10 107:4,12,14	recall 14:25 26:15	54:24 55:2,5,8,12
provides 65:24 91:6	109:5,10 111:2	50:1	55:21 56:3,11 57:2
108:5,15,25 109:13	115:14 121:22	receipt 126:16	57:11,12,23 58:14
providing 41:6	122:4	received 89:13 91:3	62:8 63:5 64:4
64:11,13 65:16	questions 9:6,9	91:4	65:16,21 94:6,10,22
104:25	26:25 122:18,20	receives 108:5,15,25	95:10,16,18 96:20
proximity 53:11	123:2,25 124:16	109:13,17,20	104:24 105:17
56:13	128:8	recess 60:23 98:5	112:6 113:2
ptab 104:8	quickly 32:18	113:16	registers 77:14
public 7:20 128:21	100:19 101:5	recited 62:6	94:13,15,18
publications 21:9	quite 119:17 123:5	recognize 61:9	regular 17:21,22
123:6	quoting 72:20	121:16	relate 30:1
publicly 18:9,11	r	recognized 120:10	related 7:21 15:8
publish 16:12 18:15	r 64:14 127:3,3	recollection 12:25	25:2,5,8,10,12,16
28:7	racing 46:25	reconstruction 31:6	26:4,11 30:6 31:10
publishable 28:10	racing 40.23	record 7:5,18 9:1	31:19 90:9 112:5
published 16:7,9	75:19 117:9,12,17	28:12 37:5 60:21,24	123:7
20:17 30:6,9 123:14	117:21 119:16	70:15 98:3,7 104:16	relates 32:13 99:8
	117.21 117.10	113:13,14,17	111:24 116:18

[relating - seen] Page 15

relating 78:4 81:24	resources 64:9	39:15 41:21 44:9,16	says 19:5 26:12
relationship 60:3	97:13,17 122:16	47:24 48:7,15 49:9	30:16 40:19,20 41:3
relative 125:13	respect 11:17 12:4	50:5 53:22 54:23	41:17,17 64:22 66:7
relevance 15:12	12:14 14:12 79:1	57:20 58:5,9,18	66:11 68:19 72:19
41:13 43:8,17 108:7	85:9,14 103:10	59:6 60:9 61:9 62:4	72:20,20 73:10 77:1
108:17 109:2,15	109:25 116:24	62:9,11,17 63:5	86:23 93:8 97:16
120:16 122:5	response 15:2,10,15	67:13 68:1,4 69:9	107:10 116:22
relevant 26:14	15:18,19 73:13 87:4	69:20 70:6,19 72:19	117:5,17 118:3,8
78:22 80:15 82:2	103:18,20,22	73:5 74:8,16,21	119:3
114:5	responsible 67:4	82:22 87:9 88:3,11	scheduler 112:4,11
relying 111:7	result 114:4	92:13 95:18,21,24	113:8
remark 22:8	retained 124:21	96:3,8,8,13,15 99:6	schematic 52:20
remember 36:2,4	retape 51:22	99:17,23 100:5,10	science 17:11
53:13 57:21 123:2,5	retrieve 54:4	100:12 101:20	scope 79:13 81:13
123:24,24	retrieved 67:14 68:3	103:6,11 104:3	82:24 96:18
remind 50:3	68:21 69:8 89:12	107:4 110:13 111:1	score 97:11
render 42:10	return 126:14	111:2,14,22 114:12	second 8:21 28:17
rendering 31:7	review 14:22 22:17	115:16,19 118:5	36:16 70:14 92:15
renders 62:2	103:9,21	122:17 124:14	98:7 121:8,9
repeat 30:5 39:5	reviewed 14:6 22:9	rim 32:14	section 21:8,12,17
65:5 95:7 109:7,10	22:12 103:12	robert 3:5 8:5	23:16 33:18,25 70:7
repeated 91:2	rgb 74:3,20	root 37:13	84:22 85:15 115:12
repeating 72:19	rich 14:9 61:8,11	routing 102:22	see 17:2 21:10,18,23
report 39:17 49:12	62:2,4,13,15,17,20	rpluta 3:10	21:24 22:11 29:8
50:13 63:23 64:1	62:23 63:8,15 64:6	rules 8:24	30:16 33:1,23 35:14
77:23 83:24 84:1,3	64:22 66:2,7,23	run 57:19,25 58:4,9	36:13 37:3,17 38:15
86:20 103:7	67:14,21 68:2,20	58:17 59:6,7	40:17 41:10 52:19
reporter 2:19 7:9	69:7,19,22 70:3,7	S	54:19 63:23 64:10
9:2 10:7 11:4 15:23	70:16,24 71:24 72:7	sample 89:18	68:23 71:5,11 72:13
36:8 40:12 61:3,16	72:16,19,20,22 73:7	sample 89:12 91:3	73:6 74:14 76:2,18
64:13 65:3 84:7	74:22,24 75:12,24	satisfactory 63:24	77:11 78:14,23
98:18,20 102:11	76:21 77:3 115:4,9	satisfies 64:16	80:18 81:5 84:12,24
104:2,4 105:23	118:3 119:1 121:16	save 56:5,5	85:17,21 87:22
123:19 125:2	122:1,13 124:7,8,12	saw 72:1	90:15 93:6 99:16
reports 23:16,20	richardson 32:14	saying 15:15 31:8	100:20 105:6 107:4
120:20,23	right 8:22 10:12	48:21,22 49:3,4,13	107:6 110:10,11
reproduce 85:24	11:8,15,17,20,25	49:19 57:6,24 59:25	113:7 114:8,15
request 7:10	12:3 13:1 14:14	62:12 63:3 64:4	115:17 116:2
require 80:9	16:3,5 17:8 18:4,13	65:19 66:2,23 73:14	117:10,19 118:22
required 79:23 80:2	19:9 20:10 21:6,14	77:22 86:17 91:18	119:8 121:11,19
requirement 80:7	21:15 22:21,22,25	94:11,23 106:6,9,9	seeing 41:15
106:4	23:1,6,10 24:1 25:5	110:16,25 113:6	seeking 19:12
requires 80:16	27:6 29:24 31:25	119:10	seen 15:9,18 16:24
research 20:20 21:1	32:7 33:13,16,20		104:11
42:3	34:14 36:5,24,25		

[select - storage] Page 16

select 102:13 107:21	89:5,25 90:9,12,13	simultaneous 42:12	spanning 21:13
107:22	90:22,23 91:10,12	simultaneously	23:11
selected 22:18	91:19,21 92:1,6,8	65:11	spans 31:24 33:12
104:22 105:5	92:10,11,12,13,17	single 33:19 53:16	85:19
selecting 87:3	92:21 93:9,10,19,23	66:14 101:7,8	speak 9:2,3
107:19,25	94:2,8,12,23 95:2,4	sir 8:19 10:9 15:25	speaking 42:12
selector 108:21	95:17 96:1,10,23	84:9 87:9 93:13,13	speaks 79:7
selects 106:19	97:10 111:16,25	93:17 98:11	special 21:22,24
sensitive 7:18	112:15,24 119:25	six 22:21 72:10	22:1,2,4,5
sentence 35:11 37:1	120:12,25	size 73:3 74:2,9,11	specialized 95:9
37:24 38:9 40:20	shaders 37:9,9	74:18	specific 34:24 37:25
41:3 68:19 70:14	120:20	sized 76:11	115:12 119:10
71:7 101:7,8 104:14	shape 26:8,9	skeptical 43:15	specifically 19:17
115:21 119:10	shapes 26:4	skgf.com 4:11,12	68:9 119:19
121:9	share 65:12	skill 75:17 78:4,25	specification 102:2
sentences 116:16	sheet 126:7,10,12,15	80:1 82:13 83:10	specified 89:19
september 1:17 2:18	128:11	114:6,11,19	speculation 42:14
7:1,6 125:18	shell 60:15	skipping 41:3	spend 13:10
sequencer 62:9	shocked 51:15	slope 71:18	square 37:13
66:24 104:25	short 20:10 37:14	sm4.0 40:21	stages 41:8
105:18 112:2,9	shorthand 2:19	small 71:3,24 75:25	standard 77:16,17
113:3,6	125:1,9	smartphone 45:5,21	80:2
sequencers 88:24	show 75:14 86:16	society 22:14,14	standards 38:11
111:20	97:8	software 47:14	78:11 79:4,17,20,20
sequences 31:10	shown 34:10,15	99:15,18	start 18:23 19:1
112:9	88:11	solution 48:20 51:16	25:23 28:6
sequencing 58:24	shows 62:15 92:12	solutions 1:21 7:10	started 93:17
67:4 113:4	sign 126:9	47:20 48:22 49:4,14	starting 21:13,16
sequentiality 119:9	signal 20:8 87:5	51:24 124:22	22:23 23:11,15 33:7
sequentially 115:5	108:2	solve 48:24	63:9 85:13
115:10,17 116:24	signals 58:12 59:10	somebody 18:5 95:5	starts 21:9 31:22
118:9 119:4,11	59:13	118:11,12	37:2 121:9
121:18	signature 10:21	sophisticated 108:1	state 7:24 48:10
service 22:14 63:21	11:12 125:21	sorry 13:22,23 20:4	125:2 126:6
set 41:7 125:5	128:14	34:12 39:5,12 54:11	statement 22:3
seven 72:10	signed 10:24 11:14	57:15 65:4 85:6,8	30:25 35:16 68:25
shader 5:24 26:21	12:15	90:4 103:18 105:23	69:2 114:13
38:23 39:1,11,15,25	significant 40:25	105:24 121:23	states 1:1 2:1 6:4,7
40:15 41:23,25	signifies 19:9	sort 37:1 48:8 62:6	6:10,13
42:19 43:6,21 44:14	signing 126:11	source 34:24 35:2,5	steps 112:10
44:16 46:3,10,20	simd 19:17 28:8	35:7 94:13 95:10	sterne 4:4 8:2,4
47:13,17 49:4 50:17	30:22,25 123:17,20	sources 34:19	stop 27:14
50:22 51:13,17,20	similar 55:16	south 2:16 3:7	storage 52:15 53:17
52:1 62:7 87:20	simple 47:3 107:20	space 126:7	54:12,23 71:1
88:1,2,8,14,16,21,22	107:25		
	1		

[store - today's] Page 17

			1
store 46:24 54:4	sure 9:8 20:9 26:3	tdutton 4:12	thing 9:16 50:23
69:19 71:25 75:6,12	27:1,6 30:3 31:23	teach 69:22	51:2 53:22 55:12
77:3,7 92:20 93:5,8	37:21,21 39:6 57:18	teaches 62:4 71:24	56:22 59:24 60:9,12
stored 71:9 72:8,17	60:20 82:10 84:1	73:7	things 17:1 30:12
73:7 74:23 75:18	88:20 89:10 97:15	team 12:21	47:25 57:3,12 73:3
76:22 124:9,12	101:15 109:10	technical 23:16,20	77:7,11 92:6 93:21
stores 52:13 53:12	switch 97:16	techniques 26:6	97:22
53:14 55:18 56:12	switching 33:4 52:6	78:13 79:6	think 11:20,25
95:8	97:18 103:8 115:2	technologies 1:8 2:8	12:20,21 14:3 16:8
storing 53:23 55:16	sworn 8:10,14	7:14 8:3 19:3,4	17:2 30:1 32:19,21
69:23	128:16	32:20 46:21 123:16	45:20 46:6 47:6
strategy 29:1	symposium 30:10	126:1 127:1 128:1	54:10 56:10,12 57:3
street 1:21 3:15	30:17	technology 19:2,13	58:19 59:3 62:11
strike 14:13 32:4	system 20:12,17,20	33:8,11,19 34:4,7	63:3 67:25 69:11
56:20 62:12 93:2	40:21 99:9 100:19	38:4 67:20 71:18	70:18 71:14 77:6
99:24	102:18 115:9,25	72:2,12 122:14	79:7 86:11,13,21
strongly 19:20,22	116:13,19 117:8	telephone 13:3	101:12,14 103:2,11
structure 30:18 57:7	121:25	tell 26:1 40:5,8,9	111:10 117:18
65:14 93:4 106:7	systems 19:8 76:13	49:2,16 52:20 97:18	119:6 122:12
108:15,25 109:13	78:8	101:8 120:5	third 28:18 116:4,5
111:6	t	telling 55:1	116:10 121:8
structures 93:1	t 127:3	term 56:15,19	thirty 126:16
struggling 47:24	take 7:17 9:12,18	terminologies	thought 54:9
student 28:24	26:1 27:5 64:8	123:25	thread 86:22 110:7
students 29:23	97:24	terminology 124:4	110:12,15,23
stuff 106:2,8,11	taken 2:15 125:4	terms 47:17,18	three 30:8 31:11,14
style 29:7	takes 22:17,19	48:23 53:23,24	62:6 83:19
subject 126:11	talk 17:21 18:6	54:15,17,19 55:16	time 7:5,24 8:21,25
submit 72:11	48:22 51:5 52:6	65:16 88:24 90:11	9:6,13 12:12 13:5,8
submitted 10:1,11	59:21 63:8 98:13	91:22 97:20 108:18	13:10 16:12 20:19
11:7 13:9 16:10	114:18	111:19	22:17,19 27:5 38:17
99:21	talked 120:24	testified 8:15	38:21,25 39:7 41:16
subscribed 125:16	talking 17:25 61:21	testifying 9:21 125:7	52:2 55:17 56:16
128:16	67:21 70:8 89:8	testimony 42:7,8	60:19,22,25 72:23
subset 118:17	93:1,17 100:18	60:16 88:19 98:14	73:17 76:5 77:21
substance 98:14	102:4 103:4	124:3,20	79:16 98:3,8 113:15
128:10	talks 67:18 86:25	textbook 59:23	113:18 114:5,25
subtract 57:17	87:3 97:13 101:14	textbooks 57:9	120:4 122:18
sufficient 97:10	tangentially 26:3	59:20,25	124:16,19,23 125:5
suggesting 47:8	tape 97:25	texture 87:18	times 91:1,2
suite 1:21 2:16	target 46:8,11	thank 8:9 60:17	titled 40:15
summary 17:19	targeted 46:1	61:6 67:10 85:11	today 9:22 14:22
18:17 22:1,18	tcb 87:5 110:24,25	100:11 122:8	43:6
support 31:21 38:12	.,20	thereof 78:20	today's 14:5 124:19
78:9		125:11	

[top - vertex] Page 18

top 33:15 107:5	43:10,20 44:6,12,19		91:21 93:23 94:17
topics 52:6	47:22 48:13 49:17	u	94:21 96:21 111:15
total 13:18 14:1	49:23 50:14,21 51:3	u.s. 61:8,19 84:10	112:6 113:2,3,5
23:22 124:20	51:8 52:5 53:1,7,15	98:23	117:20,25 118:14
		uh 20:16 28:19	· · · · · · · · · · · · · · · · · · ·
trademark 1:1 2:1	54:9,21 55:19 57:5	36:17 46:16 62:14	united 1:1 2:1 6:4,7
traditionally 56:25	58:3 59:15,22 60:20	83:2,7 116:7 119:24	6:10,13
transcribed 125:10	61:4,17 63:25 64:18	ulc 1:8 2:8 7:14 8:3	units 37:10,15 40:23
transcript 126:17	65:18 66:1,8 67:23	126:1 127:1 128:1	54:23 66:6 88:4,18
126:18	68:10 69:13,17 70:2	ultimate 47:21	102:21,22,23
transcription	74:5 75:5 80:11	uncalibrated 30:19	117:25
125:11 128:7	81:22 84:8 85:8,12	undersigned 125:1	university 17:13
transformation	86:9 90:2,5,17 91:8	understand 9:6,21	18:13
37:16	91:14 94:9 95:19	15:13 43:14 44:24	university's 18:16
transformed 71:8	97:3,24 98:10,16,21	52:3 53:5 60:14,16	unreportable 42:12
74:11	102:3,25 103:23	61:21 64:2 79:10,11	use 17:4,9,24 27:10
transforming 87:16	104:3,5 106:5	79:11,12,13,14,15	37:10,14 45:24
traverses 99:14,18	107:11 108:3,10,22	80:14 82:15,18	46:14 47:1,2 54:6
trial 1:2 2:2 6:16	109:9,19 112:12	88:20 94:10 96:17	70:18,25 74:20 97:9
triangle 73:24 74:3	113:12,20 115:1,7	114:23	100:23
triangles 26:9 27:10	117:1,3 119:20		useful 26:10
27:11 74:18	120:8,22 121:24	understanding	user 70:17,18
tried 30:21 32:23	122:10,17 123:10	42:21,25 43:23 49:7	uses 101:18
trivial 71:17	124:17	68:17 72:23 92:25	usually 18:8 27:10
true 28:13 69:2	turn 10:14 11:6,10	95:3 100:21 105:12	59:21
71:23 108:23	34:9 35:8 36:15	105:13,14 106:13	
109:11	38:6 80:12 84:21	122:13	V
try 9:3,5 28:25	87:10 106:22	understood 9:10	v 126:1 127:1 128:1
63:22 87:9	113:21 116:4 121:3	unfortunately 56:16	vector 37:11
trying 9:1 26:7,21	twice 31:17	unified 26:21 38:23	vendors 38:12
31:5,14 36:22 48:24	two 10:1,2,3,3 12:14	39:1,11,14,25 41:5	venues 17:10
	13:25 15:8 17:12,15	41:24 42:19 43:6,15	verbatim 125:8
49:2,15 52:19,21		44:14,16 46:3,10,10	verify 24:9
56:5 64:2 72:24	44:13 45:19 53:6	46:20 47:13,17 48:8	veritext 1:21 7:9
79:12 95:25 96:17	54:22 56:2 57:1	49:3 50:16,22 51:13	124:22
110:3	91:18 108:20,21	51:17,20 52:1 62:7	versatile 73:16
tuesday 1:17 2:18	116:16 124:21	80:2 92:13,16 95:2	version 39:14
7:1	tyler 4:68:4	95:3 96:1,10 119:25	versus 7:14 32:6,14
tuminaro 4:5 5:5	type 74:18 100:20	120:12,20,25	32:16
8:1,1,18 10:4,8 11:2	typical 102:17	unintelligible 95:14	vertex 26:16 37:8,15
11:5,22 15:5,16,21	typically 52:14,16	95:15 105:21	67:14,22 68:3,15,21
15:24 18:25 20:2	52:22 53:2,8,16,18	unit 52:15 53:17	69:8,19,23 70:3,5
24:4,20 25:21 27:18	55:2,5	56:18,20 62:8 64:25	71:8,25 72:5,7,16
27:24 29:16 32:10	typo 16:23,24 21:23	65:8 66:3 67:1,3	73:7,16,19 74:12,23
33:24 36:3,6,9,22	107:13	88:21 89:11,24	75:1,7 76:13,22,25
39:3,18 40:10,13		90:14 91:1,3,10,17	77:1,12,19 87:8,15
41:19 42:5,17 43:4		70.17 71.1,3,10,17	11.1,12,17 01.0,13
	T7 '4 4 T	val Solutions	1

[vertex - zhu] Page 19

104:19,21 105:1,20	wants 77:11	117:2 119:6 120:3	69:3 71:6,12 74:15
115:4,9 119:16	washington 1:22	120:15,17 121:23	74:24 75:23 76:3
121:17,25 122:2	3:16 4:9	122:8,19,22 123:11	80:6 81:6 84:5,25
124:8,11	way 7:22 17:18	123:20 125:15	100:9,15 101:17
vertexes 72:3	23:12 26:15 28:14	126:3	113:22,24 115:20
vertices 73:25,25	33:12 39:23 42:6	witnesses 125:6	116:3 121:5,20
74:11 77:10 87:17	43:23 46:12 47:15	wondering 103:13	124:6,10
97:16	48:14 49:15 51:4	word 33:15 44:24	years 71:22 72:10
video 7:6,16 122:24	52:1 54:2 57:8 60:3	45:25 97:9 100:20	72:10,25 78:21
videographer 4:14	75:4 85:19 91:20	101:3,4,9,19 102:7	yep 107:3
7:4 8:9 60:21,24	107:19 118:7	103:6 106:14	york 4:8
98:2,6 113:14,17	119:14	115:17	Z
124:18	we've 60:18	words 113:7	zhu 3:13 8:7,7 12:20
videotaped 1:15	web 34:20	work 5:21 20:3,5,15	Zitu 5.15 0.7,7 12.20
2:14	webpage 18:16	20:23 28:7,8 29:12	
view 26:23 28:22	website 18:18	36:12 42:8,15 43:16	
44:4 54:5 55:18	weeks 57:1	76:13 120:19	
75:12 105:14	weird 18:4	123:16	
vitae 5:16	welcome 61:5 98:11	worked 18:20,22	
volatile 54:18	went 82:14	19:10,18 20:7,12	
volatility 54:17	whereof 125:15	32:6	
volume 1:18 2:15	whispers 7:19	working 13:18	
5:4	wise 47:4,5	28:24 78:6,21 79:22	
voxel 30:7,8	witness 5:2 8:10	79:22 87:6 118:19	
vs 1:7 2:7	11:20 15:4,13 18:22	119:15	
W	20:1 24:3 25:20	works 26:18 50:3	
wacker 3:7	27:17,23 29:14 32:9	write 29:5 54:13	
want 16:14,18 17:1	33:22 36:1 38:25	writing 21:25 29:6	
24:12 27:4 30:24	39:17 40:3,5 41:15	29:22 54:15 55:16	
41:17 46:14,22,23	42:2,15 43:2,9,18	written 21:3 80:25	
46:23 47:1,2 51:11	43:25 44:11,18	119:8	
52:2 81:14 82:15	47:12 48:12 49:12	wrong 63:2	
83:19 86:6,12 87:24	50:13,20 51:1,15	wrote 22:3,7 37:24	
88:20 89:1,7 90:10	52:25 53:5,11 54:11	X	
90:13,18,19,19	55:15 56:25 59:9,19	x 3:13	
93:20 96:18,23	63:18 64:14 65:4,24	xenos 120:10	
100:5 101:15 106:3	66:6 67:17 68:6	y	•
106:11 107:17,24	69:11,14 70:1 73:21		
115:13 118:16,25	75:3 80:6 81:21	yeah 12:18 13:7 14:3 15:17 17:20	
119:7	86:6 90:8,25 94:5	20:1 21:21 31:23	
wanted 28:11 96:2	95:1 97:1 101:25	35:3 36:14 38:8	
96:11	102:12 105:24	40:18 45:15 48:12	
wanting 48:18	107:9,17 108:8,19		
	109:3,16 112:2	54:11 55:22 56:6	
	114:22 116:22	58:8,19 60:20,20,20	

Federal Rules of Civil Procedure Rule 30

- (e) Review By the Witness; Changes.
- (1) Review; Statement of Changes. On request by the deponent or a party before the deposition is completed, the deponent must be allowed 30 days after being notified by the officer that the transcript or recording is available in which:
- (A) to review the transcript or recording; and
- (B) if there are changes in form or substance, to sign a statement listing the changes and the reasons for making them.
- (2) Changes Indicated in the Officer's Certificate. The officer must note in the certificate prescribed by Rule 30(f)(1) whether a review was requested and, if so, must attach any changes the deponent makes during the 30-day period.

DISCLAIMER: THE FOREGOING FEDERAL PROCEDURE RULES

ARE PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

THE ABOVE RULES ARE CURRENT AS OF SEPTEMBER 1,

2014. PLEASE REFER TO THE APPLICABLE FEDERAL RULES

OF CIVIL PROCEDURE FOR UP-TO-DATE INFORMATION.

Page 1

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

LG ELECTRONICS, INC.)
Petitioner,)
VS.) No. Ipr 2015-00325
ATI TECHNOLOGIES ULC,)
Respondent.)

The deposition of NADER BAGHERZADEH, taken before JO ANN LOSOYA, C.S.R., pursuant to the provisions of the Illinois Code of Civil Procedure and the Rules of the Supreme Court thereof pertaining to the taking of depositions for the purpose of discovery at 71 South Wacker Drive, Chicago, Illinois commencing at 9:05 a.m. on August 14, 2015.

ATI 2074 LG v. ATI IPR2015-00326

		Dago	2	Dago 4
	DD EGGD VII	Page		Page 4
1 2	PRESENT: MAYER BROWN ROWE & MAW		1	
3	MR. ROBERT G. PLUTA		2	\mathcal{E}
3	MR. JOHN X. ZHU 71 South Wacker Drive		3	
4	Chicago, Illinois 60606-4637 (312) 701-8641		4	<i>Dy</i> 1.11. 1 <i>m</i> 1111111111
5	rpluta@mayerbrown.com		5	-
6	rzhu@mayerbrown.com Appeared on behalf of the Petitioner;		6	<i>= 100 = 20</i>
7	Appeared on benan of the Fethioner,		7	
8	STERNE KESSLER GOLDSTEIN FOX MR. JONATHAN TUMINARO, Ph.D.		8	
	MR. ZHU HE		9	· ·
9	1100 New York Avenue NW Washington, DC 20005		10	1
10	(202) 371-2600		11	Emiliar (o. 1
11	jtuminar@skgf.com zhe@skgf.com		12	
	Appeared on behalf of the Respondent.		13	
12	ROBIN KAPLAN		14	E/MIOTE 1 (0. 1
13	MR. BRYAN J. MECHELL		15	Exhibit No. 5
14	800 LaSalle Avenue Suite 2800		16	Exhibit No. 6
	Minneapolis, Minnesota 55402		17	Exhibit No. 7 69 24
15	(612) 349-0172 bmechell@robinskaplan.com		18	Exhibit No. 8 93 12
16	Appeared on behalf of the Respondent.		19	Exhibit No. 9 113 17
17 18	ALSO PRESENT:		20	Exhibit No. 10 148 8
19	Mary Ann Naas, Videographer.		21	-
20 21			22	2
22 23	DEDORTED DV. IO ANNI I OSOVA		23	3
24	REPORTED BY: JO ANN LOSOYA LICENSE #: 084-002437		2 4	1
		Page	3	Page 5
1				THE VIDEOGRAPHER: We are on record. My
2			,	name is Mary Ann Naas representing Veritext.
3				Today's date is 8-14-2015. The time is
4				4 approximately 9:07 a.m.
5			ĺ	This deposition is being held at
6				Mayer Brown located at 71 South Wacker Drive,
7			.	7 Chicago, Illinois, and is taken by the patent owner.
8				The caption of this case is LG Electronics, Inc.,
9				versus ATI Technologies ULC. This case is being
10			10	· · · · · · · · · · · · · · · · · · ·
11			11	
12			12	Nader Bagherzadeh.
13			13	
14			1	· · · · · · · · · · · · · · · · · · ·
15			15	
16			16	
17			1.	i i i i i i i i i i i i i i i i i i i
18			18	
19			119	
20			20	-
21			2:	
22			22	
23			23	
			2	
24			12'	IVIN. ZITO. JOHN ZHU HOIH WAYEL DIOWH AISO

	Page 6		Page 8
1	on behalf of LG Electronics.	1	under oath here this morning?
2	THE VIDEOGRAPHER: Our court reporter,	2	A. Absolutely.
3	JoAnn Losoya, representing Veritext, will now swear	3	Q. And all day today actually.
4	in the witness.	4	A. Absolutely.
5	WHEREUPON:	5	Q. Is there any reason you cannot do that?
6	NADER BAGHERZADEH,	6	A. No, I cannot see a reason why I could not
7	called as a witness herein, having been first duly	7	do that.
8	sworn, was examined and testified as follows:	8	Q. Okay. Do you know why you're here today?
9	EXAMINATION	9	A. Yes.
10	BY MR. TUMINARO:	10	Q. Why?
11	Q. Good morning. Please state your name,	11	A. There's a case pending.
12	sir.	12	Q. And you submitted a declaration in that
13	A. Nader Bagherzadeh.	13	case?
14	Q. And please state your home address?	14	A. Yes. If you have it, I can show it to
15	A. My address is 415 Hilledge in Laguna	15	you.
16	Beach, California.	16	(WHEREUPON, document marked as
17	O. Your work address?	17	Exhibit No. 1.)
18	A. University of California, Irvine, Irvine,	18	BY MR. TUMINARO:
19	California.	19	Q. Sir, you've been handed what has been
20	Q. Sir, I understand that you have been	20	marked as Exhibit 1. This is titled Declaration of
21	deposed several times before; is that right?	21	Dr. Nader Bagherzadeh. Is this your declaration?
22	A. That's correct.	22	A. Yes, sir.
23	Q. Just a couple of ground rules for this	23	Q. If you turn to the last page of the
24	deposition. As you know, we're trying to get a	24	declaration before the appendix, there's a signature
	Page 7		Page 9
1	clear record. So I'll ask that you don't speak over	1	there?
2	me, and I'll try not to speak over you. Is that	2	A. Yes.
3	fair?	3	Q. Is that your signature?
4	A. That is fair.	4	A. Yes.
5	Q. Okay. I'm going to try to ask clear	5	Q. And you signed this declaration on
6	questions; but if at any point you don't understand	6	December 9, 2014?
7	my question, will you let me know?	7	A. Yes.
8	A. Sure. Can I ask you to slow down a	8	Q. Is there anything that you'd like to add
9	little bit?	9	to this declaration?
10	Q. Oh, sure. Sure. Okay.	10	A. No.
11	If you answer one of my questions,	11	MR. PLUTA: Counsel, did you I thought
12	I'm going to assume that you understood it. Is that	12	I heard you refer to this as Exhibit 1. Are you
13	fair?	13	referring to it as a different exhibit number?
14	A. Yes. If I answer, that means I	14	MR. TUMINARO: Just Exhibit 1 to this
15	understood it, correct.	15	deposition.
16	Q. Okay. I'm going to take periodic breaks,	16	MR. PLUTA: Okay. So you are not going
17	but if at any time you need a break, would you let	17	to refer to it as Exhibit 1003.
18	me know?	18	MR. TUMINARO: I may during the course of
19	A. Absolutely.	19	it. I just wanted to mark this so that it is clear
20	Q. Okay. But if there's a pending question,	20	this is Exhibit 1 to Dr. Bagherzadeh's deposition.
21	I'd ask that you answer the question before we take	21	MR. PLUTA: Fair enough.
22	a break. Is that fair?	22	BY MR. TUMINARO:
23	A. Sure.	23	Q. Is there anything that you would like to
24	Q. You understand that you're testifying	24	delete from your declaration?
_ 1	2. Tou understand that you're testifying		detect from your decidration:

Page 10 Page 12 1 A. No, I don't think so. 1 this. 2 2 Q. Are there any changes that you'd like to Q. But, in fact, a figure from one of those 3 make to your declaration at all? 3 lectures is actually in your declaration, right? 4 4 A. It could be. I couldn't say for certain. 5 Q. If you would turn with me to Paragraph 2 5 Q. Let's look at the next page, Page 11. In 6 of your declaration. 6 Paragraph 31, there's also a figure. Where did this 7 7 A. Okay. figure come from? 8 Q. Paragraph 2 reads: "I have been asked to 8 A. From one of my lectures. 9 provide my opinions and views on materials I have 9 Q. Your lectures aren't listed in the 10 10 reviewed in this case related to U.S. Patent materials considered, right? 11 No. 7742053 (the '053 patent)." 11 A. It's from my knowledge that I have. 12 Do you see that? 12 Q. Well, I appreciate the answer, but I 13 13 think it didn't answer my question. This lecture is A. Yes. 14 Q. What materials are you referring to that 14 not listed on your materials considered, right? 15 15 you reviewed? MR. PLUTA: Object to form. 16 A. I think we have a list of that. If I 16 BY THE WITNESS: 17 refer you to Page 87 of Exhibit 1, you can see the 17 A. It's what I know. So the lecture is not 18 18 cited, that's right, but it's what I know. I have a 19 19 Q. That's titled Materials Considered and lot of other knowledge that is not cited here about 20 Exhibit List, correct? 20 computer architecture, the textbooks that I've used 21 A. That is correct. 21 over the years, my papers I've written over the 22 Q. Does this list all the materials that you 22 years. No, they're not cited here, but they're up 23 23 reviewed in preparing your declaration? in my head. 24 24 A. Yes. Page 11 Page 13 1 Q. In fact, there are no other materials BY MR. TUMINARO: 2 that you reviewed in preparation for your 2 Q. Is there anything else that you relied 3 declaration? 3 upon in preparing your declaration that you did not 4 If they were, I would have put them in 4 list in your exhibits considered? A. 5 5 A. I cannot think of anything right now here. 6 Q. Okay. If you would turn with me to 6 unless you point it out. 7 7 Page 10 of your declaration. Q. The figure in Paragraph 31, do you know 8 8 what lecture this refers to? A. Yes. 9 9 Q. At the top of Page 10, there's a figure A. Computer architecture. 10 or a picture. 10 Q. Is that publicly available? 11 11 Do you see that? A. No. 12 12 A. Hm-hmm. MR. TUMINARO: Counsel, I'd guess I'd 13 You didn't generate this picture, did 13 state for the record that we want a copy of that 14 14 you? lecture. It's not listed on the materials 15 15 A. It was probably from one of my lectures. considered, and it appears to be something that was 16 One of the lectures that you prepared? 16 used in generating this declaration. Q. 17 17 MR. PLUTA: Other than the figure and A. Yeah. 18 18 Q. But that's not listed on your materials what he just testified, I don't see how the 19 19 considered, is it? relevance of the lecture in its entirety would be 20 20 A. It's one of my lectures, yeah. relevant to this proceeding. 21 Q. So. One of your lectures is one of the 21 BY MR. TUMINARO: 22 22 things that you considered in preparing your Q. Did you write your declaration? 23 declaration? 23 A. Yes, sir. 24 A. My knowledge is what was considered for 24 Q. How much time did you spend in preparing

Page 14	Page 16
1 your declaration? 1 A. No. We desig	med boards.
	Bell Labs in 1984?
3 Q. Over 40 hours. Was it over 50 hours? 3 A. Yes, sir.	
	t 30 years, you have been
5 Q. When were you first contacted with 5 in academia, right?	•
6 respect to this matter? 6 A. Not accurate.	
	curate about my statement?
8 Q. Who contacted you? 8 A. Because I wer	•
	In 1987, you started in
10 the experts to the lawyers. 10 academia; is that right	?
11 (WHEREUPON, document marked as 11 A. That's right.	
	ou have been in academia,
13 BY MR. TUMINARO: 13 right?	,
	a little period of running
	have seen it in my resume.
	er actually built a computer
17 Ph.D. 17 chip, have you?	, J
Do you see that? 18 A. I did, yes.	
19 A. Yes. 19 MR. PLUTA:	Object to form.
Q. You have seen this document before? 20 BY MR. TUMINARO	· ·
21 A. Yes. 21 Q. You did build	a computer chip.
22 Q. This is your current CV? 22 A. Absolutely.	I I
23 A. Yes. 23 Q. What did you	do then?
24 Q. Just to be clear, is this just your 24 A. I designed it.	
Page 15	Page 17
1 consultant CV? It says consultant CV. 1 Q. You designed it	t?
2 A. It says so, yes. 2 A. Yes.	
	ne that just designing a
4 A. I have different CVs, yeah. 4 chip is building it?	,
5 Q. This is the one that you use for patent 5 A. Excuse me.	
1	me that just designing a
7 A. It seems that way. I mean, it's here 7 chip is building a chip?	
8 so 8 A. You didn't ask	
	gned it, and I said I designed
	design something. Then you
11 Q. Anything you want to delete? 11 have to fabricate it.	5 5 7 7 2
12 A. No. 12 Q. Have you built	a chip?
	that what you are asking
14 A. No. 14 me?	, <u>e</u>
	what the word "build" means?
	what the fabrication means?
	now what the word "build"
18 Q. While you were there, did that work 18 means?	
	use building. We use the
The state of the s	
20 A. It did. 20 word fabrication. We f	abricated the chip, ves.
21 Q. What did you do? 21 Q. You fabricated	the chip, and you designed
21 Q. What did you do? 22 A. Used them. 21 Q. You fabricated 22 the chip?	

	Page 18		Page 20
1	MR. PLUTA: Object to form.	1	BY MR. TUMINARO:
2	BY THE WITNESS:	2	Q. Your knowledge about
3	A. We designed pieces of the GPU for our	3	A. Knowledge, yes.
4	chip.	4	Q. So Exhibit 3 is an example of your
5	BY MR. TUMINARO:	5	knowledge?
6	Q. Which chip are you referring to?	6	A. That's right. That's better. Yes, thank
7	A. Multiple chips.	7	you.
8	Q. Did you build a chip called morphosys?	8	Q. If you will turn with me in Exhibit 3 to
9	MR. PLUTA: Object to form. Object to	9	Page 10-3.
10	relevance.	10	A. Hm-hmm.
11	BY THE WITNESS:	11	Q. And on the left-hand column, about midway
12	A. Yes, I did.	12	through the page, there's a heading four, Morphosys,
13	(WHEREUPON, document marked as	13	System Model.
14	Exhibit No. 3.)	14	Do you see that?
15	BY MR. TUMINARO:	15	A. Yes.
16	Q. Sir, you have been handed what has been	16	Q. And the last sentence, it says: "The
17	marked as Exhibit 3 for identification purposes. Do	17	system model and architecture details for the first
18	you recognize this document?	18	implementation of morphosys (M1 chip) are described
19	A. Yes.	19	hereafter."
20	O. What is it?	20	Do you see that?
21	A. It's a paper I have written with my	21	A. Yes, I see that.
22	colleagues and former students.	22	Q. Was this morphosys actually a chip?
23	MR. PLUTA: I'm going to object to the	23	A. Yes, sir.
24	relevance of this exhibit at this time.	24	O. So the architectural details were
24		24	
	Page 19		Page 21
1	BY MR. TUMINARO:	1	sufficient to be an implementation of a chip?
2	Q. If you turn with me to I guess what is	2	MR. PLUTA: Object to form.
3	labeled as Page 10-3. Well, first of all first	3	BY THE WITNESS:
4	of all, if you will turn to your CV again,	4	A. So your question is by looking at this
5	Exhibit 2, on Page 19 of your CV.	5	paper, do I prove that I fabricated the chip or not?
6	A. Yeah.	6	BY MR. TUMINARO:
7	Q. Are you there, sir, Page 19?	7	Q. That was not my question.
8	A. Yes.	8	A. But I'm getting that from your question.
9	Q. Okay. If you look at C57, there's a	9	I'm repeating it because I didn't understand what
10	paper there that's listed, and the title of the	10	you said.
11	paper is: "Morphosys: An Integrated Reconfigurable	11	Q. Then I'll clarify.
12	Architecture."	12	A. Right. Please.
13	Do you see that?	13	Q. In this sentence in your paper, it says:
14	A. Yes.	14	"The system model and architectural details for the
15	Q. If you look at the title of Exhibit 3,	15	first implementation of the morphosys chip are
16	it's the same, right?	16	described hereafter." Right?
17	A. Yes.	17	A. Right.
18	Q. Exhibit 3 would be an example of some of	18	Q. So, you had an implementation?
19	the stuff that's in your head as you said earlier,	19	A. I had an implementation. That's right.
· ') ()	right?	20	Q. That implementation was based on
20	MR. PLUTA: Object to form.	21	architectural details of the chip?
21			
21 22	BY THE WITNESS:	22	A. That is described in this document, yes.
21		22 23 24	A. That is described in this document, yes. Q. What were the architectural details that you had made at that time?

	Page 22		Page 24
1	MR. PLUTA: Object to form. Object to	1	architectural details are, correct?
2	relevance.	2	A. Hm-hmm.
3	BY THE WITNESS:	3	Q. You said it's described in the paper,
4	A. It's in the paper.	4	right?
5	BY MR. TUMINARO:	5	A. Yes.
6	Q. Let's look at what the paper says.	6	Q. And I went to Page 10 and the
7	A. I'll be glad to go over it with you.	7	architectural details. Now, my question is are the
8	Q. If you will turn with me to Page 10-10.	8	architectural details the VHDL that's described on
9	On the right-hand side, there's a number 7,	9	Page 10 of your paper?
10	Interactive Software Environment.	10	MR. PLUTA: Object to form. Object to
11	Do you see that?	11	relevance.
12	A. 10-10. Which column?	12	BY THE WITNESS:
13	Q. On the right-hand side.	13	A. Sir, if I read that sentence
14	A. Yes.	14	MR. PLUTA: Object to foundation.
15	O. The first sentence after Interactive	15	BY THE WITNESS:
16	Software Environment, it reads: "The morphosys	16	A. If I read that sentence, it should give
17	reconfigurable system has been specified in	17	you the information. I'll read it again. It
18	behavioral VHDL."	18	says let me finish the sentence, please.
19	Do you see that?	19	The morphosys reconfigurable system
20	A. I do.	20	has been specified in behavior VHDL. Which
21	O. What is VHDL?	21	morphosys am I talking about? It's the one that
22	A. It stands for a hardware description	22	that is discussed in this paper. So the relevance
23	language.	23	is obvious. You have to deduce it from this
24	Q. Verilog Hardware Description Language?	24	statement. If you are not getting that information,
2 1			
	Page 23		Page 25
1	A. It says VHDL. It doesn't say verilog.	1	maybe we didn't write it properly.
2	Q. Does VHDL stand for Verilog Hardware	2	BY MR. TUMINARO:
3	Description Language?	3	Q. That's what I'm asking you. I could read
4	A. No, it does not. I don't think it does.	4	the paper. I'm asking you are the architectural
5	Q. What does the V stand for, then?	5	details the VHDL?
6	A. I don't know. It could be.	6	MR. PLUTA: Object to form. Object to
7	Q. Having read that sentence, turning back	7	relevance. Object to foundation.
8	to the sentence on Page 3 that talked about	8	BY THE WITNESS:
9	architectural details, were the architectural	9	A. I repeat my at the beginning, we talk
10	details that are described on Page 3 the VHDL code	10	about architectural details. This is referring to
11	that you were talking about?	11	the morphosys reconfigurable system. Which other
12	MR. PLUTA: Object to form. Object to	12	architecture are we talking about if it's not what I
13	relevance.	13	just said in Page 2 that you had? Can you tell me
14	BY THE WITNESS:	14	that I could be talking about something else? I
15	A. Well, the statement there is accurate.	15	just don't see why it's so puzzling to you.
16	It says the morphosys reconfigurable system has been	16	BY MR. TUMINARO:
17	specified in behavior VHDL.	17	Q. Is the answer to my question, yes, then,
18	BY MR. TUMINARO:	18	that the architectural details are the VHDL?
19	Q. You didn't answer my question. On Page	19	MR. PLUTA: Object to form. Object to
20	3, you said the system the system model and	20	relevance.
21	architectural details for the implementation of	21	BY THE WITNESS:
22	morphosys (M1 chip) are described hereafter, right?	22	A. It says morphosys reconfigurable system
	A Vou road that Voc	23	has been specified in VHDL. It could not be any
23	A. You read that. Yes.Q. I asked you earlier about what	24	other morphosys circuit.

Page 26 Page 28 1 BY MR. TUMINARO: 1 BY THE WITNESS: 2 2 Q. So that means that the VHDL was an A. My resume speaks for itself. I never 3 3 implementation of that morphosys chip? 4 MR. PLUTA: Object to form. Object to 4 BY MR. TUMINARO: 5 5 Q. You never programmed a 3D graphics on a relevance. Asked and answered. 6 6 BY THE WITNESS: GPU, did you? 7 7 A. Yes. That's what it says. Of course. A. I taught a course, and I did some 8 8 programming in open GL for the course to prepare the BY MR. TUMINARO: 9 Q. If you turn back with me to your 9 exams, the homework, and so on. Yeah. 10 10 Q. Have you ever used a -- strike that. You declaration at Paragraph 38, I'll read it for the 11 record. Are you there? 11 mentioned the open GL. Have you heard of DX10, 11? 12 A. Yeah. 12 A. Yes. I have heard of those. 13 13 Q. Paragraph 38 reads: "I believe that a Q. Have you ever used those? 14 14 person of ordinary skill in the art relating to the A. Not specifically, but I have -- I am 15 '053 patent would be someone with a good working 15 supervising projects or supervised projects, we 16 knowledge of computer graphic processing 16 tried to use GPUs for high performance computing and 17 architecture as well as the systems and programs 17 we're still doing that. 18 18 Q. By high performance computing, you don't that support such architecture." 19 19 mean 3D graphics, do you? Do you see that? 20 A. Yes. 20 A. Well, you're using the GPUs, but you are 21 Q. How did you come up with that definition? 21 doing it for number crunching. 22 A. I just wrote it. 22 Q. Right. 23 O. Based on what? 23 A. It's called GPGPUs. 2.4 A. Based on my knowledge. 24 Q. It's general purpose GPU, right? Page 27 Page 29 1 Q. Do you satisfy that definition? 1 A. Yes. But we're using all the resources 2 A. It's in the report. So I'm happy with 2 on the GPU. All the resources. 3 3 it. Q. Okay. My question is you are not using 4 Q. No, no. That wasn't my question. 4 the GPU to do 3D graphics processing, right? 5 5 Are you a person of ordinary skill in A. Not now. 6 the art with respect to your own definition? 6 Q. Have you ever? 7 7 A. Sorry about that. I took it as satisfied A. Before we did, yes. 8 8 differently. Q. If you look back with me, I guess, at 9 9 Paragraph 31, there's that figure that you put in Yes, I am. 10 10 Q. Were you a person of ordinary skill in from your lecture. 11 the art as of the time that the '053 patent was 11 A. Yes. 12 12 filed? Q. That figure is relating to a 13 A. Yes, I was. 13 multi-threaded CPU, right? 14 14 Q. Have you ever designed a GPU? A. It's related to multi-threading in 15 15 A. I designed some of the architectures that general. And you could think about it as a CPU. 16 we have that were used for computer graphics, yes. 16 You could think about it as a GPU. You can think 17 Q. Architectures other than the morphosys? 17 about it as any computation form. It's not target 18 18 A. Yes. I have done several architectures, application specific. 19 19 but morphosys was used for graphics. Q. If you look at the previous page, 20 Q. That had graphics in it? 20 Page 10, that figure is for multi-threading on a 21 A. Yes. 21 CPU, right? 22 22 Q. You never worked in a graphics company, A. It says CPU, right. I couldn't tell you 23 23 did you? that's a GPU. It says CPU. 24 MR. PLUTA: Object to form. 24 Q. The '053 patent is about a GPU, not a

Page 30 Page 32 1 CPU? So I'm going to look for references to see if there 1 2 2 MR. PLUTA: Object to form. are. 3 BY THE WITNESS: 3 Q. Okay. Great. 4 A. '053 is about multi-threading for GPUs. 4 If I'm misunderstanding, please let me A. 5 5 So this is to give you information about the know. 6 background of multi-threading. It's not talking 6 It doesn't seem to be. 7 Q. Where does this material come from then? 7 about -- I'm not trying to address GPUs here. I'm 8 8 A. I think I addressed that earlier. From talking about what is multi-threading for the layman 9 9 my knowledge and lectures, research, all of it done 10 10 BY MR. TUMINARO: over the years. 11 11 Q. So the entirety of the technical Q. Let's go back to your materials 12 considered. 12 background section just comes from stuff that you 13 13 have in your head? A. Yes, sir. 14 Q. Just to be clear, your lectures aren't 14 MR. PLUTA: Object to form. 15 shown in the materials considered, right? 15 BY THE WITNESS: A. Yes. That's why I'm here. 25 years of 16 MR. PLUTA: Object to form. Asked and 16 17 answered. 17 experience in education. If you want me to cite 18 18 BY THE WITNESS: everything I know, then it will be a book. 19 19 A. I think we established that, Counsel. BY MR. TUMINARO: 20 BY MR. TUMINARO: 20 Q. If you'd turn with me to Page 8, I'd like 21 Q. Is the answer yes? 21 to focus on the Bullet No. 2 that says 22 A. It's clear that it's not. 22 rasterization. 23 23 Q. Earlier you mentioned that you had 24 designed GPUs or certain aspects of GPUs for your 24 O. Rasterization, that's about generating Page 31 Page 33 1 chips. Those previous designs aren't listed in your pixels; is that right? 2 previous materials considered? 2 A. Yep. 3 3 MR. PLUTA: Object to form. Q. If you'd turn with me to Page 10, the 4 BY THE WITNESS: 4 last bullet there talks about cache misses. 5 5 There's a penalty for a cache miss, A. No, sir. 6 BY MR. TUMINARO: right? 6 7 7 Q. Let's go back to your technical A. Yes. 8 background which starts on Page 7 of your 8 Q. So if you increase the size of the cache, 9 9 declaration. And the technical background spans that will likely decrease cache misses; right? 10 from Page 7 through to the top of Page 12, correct? 10 A. There is a reason for that, but you 11 11 A. Yes. cannot assume increasing the cache size will always 12 12 Q. And in that entire section, there's not a improve your performance. There is a limit to that. 13 single cite to any external material, correct? 13 This is what is called a bathtub curve. 14 14 A. Let me check. Q. Bathtub curve, you moved your hand --15 15 MR. PLUTA: Object to form. A. Just like a bathtub. You cannot keep 16 BY MR. TUMINARO: 16 increasing the cache size and expect improvements, 17 17 but it gets better. You're right. Q. Let me ask the question again. 18 18 In that entire section, you didn't Q. And by -- and you said increasing the 19 19 cite any extrinsic evidence to support any statement cache size will increase your performance? 20 20 that you made in this section, right? A. Yeah. 21 A. Would you like me to check? I was just 21 Q. By increased performance, you mean 22 22 about to do that for you. decreased cache misses? 23 23 Q. Yeah. I wanted to clarify my question. A. It will give you more room to put stuff 24 A. Yes. I think I understood your question. 24 in there so you can keep track of more instructions

related to general topics. is the cache block. It's all related. Q. Maybe my question is not perfect. So, a cache miss is a penalty, right? A timing penalty? A. Yes. You're right. Q. Yes. You're right. Q. Fly our miss in the cache, you have to go out and get whatever it is you're looking for in memory, right? A. Then it is right, and penalty in prower comments. The memory in the memory, that's right. Q. So it's a penalty in time? Q. Great. So if you decrease the cache size, that will likely increase cache misses; right? MR. PLUTA: Object to form. Page 35 The WITNESS: A. You dian't qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs. let's talk about GPUs. A. You don't qualify your question with that. You asked me a general question. Q. Oxay. How about in GPUs. A. You dan't qualification is still not sufficient for me to answer that. But let me help you out. If you want metals a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. Alarger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses; make the cache size, will increase cache misses? MR. PLUTA: Object to form. Page 35 Fage 37 Fashibit No. 4.) The WITNESS: A. You dian't qualify your question with that. You asked me a general question. Q. Oxay. How about in GPUs; If you decrease the cache size, will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. Page 35 A. You dian't qualify your question with that. You asked me a general question. Q. Oxay. How about in GPUs; If you application is still not sufficient for me to a likely day out question with that. You asked me a general question. Q. Oxay. How about in GPUs; If you decrease the cache size, will likely increase the likelihood of a cache miss in a GPU, right? A. You dian't qualify your question with that. You didn't qualify your question with that you decrease the cache size, wi		Page 34		Page 36
2	1	or data, and then there's something called how big	1	related to general topics.
Q. Maybe my question is not perfect. So, a cache miss is a penalty, right? A triming penalty? 4			2	
cache miss is a penalty, right? A timing penalty? A. Yes. You're right. Q. I'you miss in the cache, you have to go out and get whatever it is you're looking for in memory, right? A. From the slower memory, that's right. Q. So it's a penalty in time? A. That is right, and penalty in power consumption. Q. Great. So if you decrease the cache size, that will likely increase cache misses; right? MR. PLUTA: Object to form. Page 35 That. So you have to qualify your question is for what type of application you're talking about. Q. May. How about in GPUs. If you have that. You asked me a general question. Q. Al right increase the cache size will likely horease the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. Page 35 A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs! If you are decrease the cache size will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. A. You can write two lines of GPUs, and it a larger program, yes, you need a bigger cache. MR. PLUTA: Object to form. A. You all right. Since his case is about decrease the cache size will linerease cache. MR. PLUTA: Object to form. A. You can write two lines of GPUs, and it a larger program, yes, you need a bigger cache. MR. PLUTA: Object to form. MR. PLUTA: Object to form. A. You can write two lines of GPUs, and it a larger program, yes, you need a bigger cache. MR. PLUTA: Object to form. MR. PLUTA			3	
A. Yes. You're right. Q. If you miss in the cache, you have to go out and get whatever it is you're looking for in memory, right? A. From the slower memory, that's right. Q. So it's a penalty in time? A. That is right, and penalty in power consumption. Great. So if you decrease the cache size, that will likely increase cache misses; right? A. What application do you have in mind? BY MR. PLUTA: Object to form. BY MR. TUMINARO: It will fit in a small cache. So you don't need Page 35 A. Well, if your application is three lines, it will fit in a small cache. So you don't need Page 37 A. You dan't to qualify your question with that. You asked me a general question. Q. All right. Since this case is about decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it a larger program, yes, you need a bigger cache. See MR. PLUTA: Object to form. BY MR. TUMINARO: A. Well a going? It is up to you, sir. THE WITNESS: In okay. If I get tired, THE WITNESS: In okay. If I guestion like you said. MR. TUMINARO: (WHEREUPON, document marked as Exhibit No. 4.) BY MR. TUMINARO: L. Was Exhibit 4 for identifications purposes. This is U.S. Patent No. 7233335. Have you seen this document before? A. You. Correct. Q. And this is listed on your materials considered; correct? A. Correct. WHEREUPON, document marked as Page 37 Exhibit No. 5.) BY MR. TUMINARO: Q. Sir, you have been handed what has been marked as Exhibit No. 5.) BY MR. TUMINARO: Q. Sir, you have been handed what has been marked as Exhibit No. 5.) BY MR. TUMINARO: Q. Sir, you have been handed what has been marked as Exhibit No. 5.) BY MR. TUMINARO: A. Correct. Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir; A. Yes. Q. And Ill read it for the record. "A Plurality of command processing engines. Do you see that? A. Yes. Pupulative of command processing engines, coupled to the arbit				
G. If you miss in the cache, you have to go out and get whatever it is you're looking for in memory, right? A. From the slower memory, that's right. Q. So it's a penalty in time? 10 Q. Great. So if you decrease the cache 11 A. That is right, and penalty in power 12 consumption. 13 Q. Great. So if you decrease the cache 14 size, that will likely increase cache misses; right? 15 MR. PLUTA: Object to form. 16 BY THE WITNESS: 17 A. What application do you have in mind? 18 BY MR. TUMINARO: 19 Q. I'm asking you in general. If you have a 10 smaller cache or fewer things in there, it is more 11 likely that you are going to have a miss a cache 12 miss, right? 10 A. Well, if your application is three lines, 11 that. So you have to qualify your question is for 12 what type of application you're talking about. 13 Q. All right. Since this case is about 14 GPUs, let's talk about GPUs. 15 A. You didn't qualify your question with 16 that. You asked me a general question. 17 Q. Okay. How about in GPUs? If you 18 decrease the cache size, it will likely increase the 19 delay would use in a GPU, right? 10 MR. PLUTA: Object to form. 11 BY THE WITNESS: 12 A. You can write two lines of GPUs, and it 13 still would be a very good size cache. So your 14 qualification is still not sufficient for me to 15 answer that. But let me help you out. If you want 16 a large program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 MR. TUMINARO: 20 All read what has been marked as Exhibit 5 for identification purposes. 21 This is the would use in a commercial implementation, of command processing engines. 22 If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? 23 A. Vesh. 24 Q. The last element in claim No. 5 is a plantation of the arbiter, each operable to receive and process the command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 24 Droy ou see that? 25 Droy ou see that? 26 Droy ou see that? 27 Droy ou see that?				-
out and get whatever it is you're looking for in memory, right? A. From the slower memory, that's right. Q. So it's a penalty in time? A. That is right, and penalty in power consumption. Q. Great. So if you decrease the cache size, that will likely increase cache misses; right? MR. PLUTA: Object to form. Page 35 that. So you have to qualify your question with that. You asked me a general question. A. You cache miss in a GPU, right? MR. TUMINARO: Okay. Great. (WHEREUPON, document marked as Exhibit 5 for identifications purposes. This is us. Page 37 Till let you know if it's not in the middle of a question like you said. MR. TUMINARO: Okay. Great. (WHEREUPON, document marked as Exhibit 15 not. 4.) BY MR. TUMINARO: Q. You have been handed what has been marked as exhibit of infectingation purposes. This is us. Page 37 have you seen this document before? A. Yes. Q. I'm asking you in general. If you have a miss—a cache miss, right? A. Well, if your application is three lines, it is more things, right? A. Well, if your application is three lines, it is more that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, and it at. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. TUMINARO: Q. Is that should be a very good size cache. So your question with that and the proportion is the middle of a qualification is still not sufficient for me to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. In this is the Morton patent, correct? A. Correct. Q. I'm sis the Morton patent, correct? A. Correct. BY MR. TUMINARO: Q. Sin this is the Morton patent has been marked as considered, correct? A. Correct. Q. I'm sis the Morton patent has been marked as lease that it 5 for identification purposes. This is U.S. Patent No. 7742053. This is the		-		
memory, right? A. From the slower memory, that's right. Q. So it's a penalty in time? A. That is right, and penalty in power consumption. Q. Great. So if you decrease the cache size, that will likely increase eache misses; right? MR. PLUTA: Object to form. BY THE WITNESS: A. What application do you have in mind? BY MR. TUMINARO: Q. To use when the slower memory that's right. A. What application do you have in mind? BY MR. TUMINARO: Q. You have been handed what has been marked as Exhibit 4 for identifications purposes. This is U.S. Patent No. 7233335. Have you seen this document before? A. Yes. Q. This is the Morton patent, correct? A. Correct. Q. And this is listed on your materials considered, correct? A. Correct. Q. And this is listed on your materials considered, correct? A. Correct. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. Do you see that? MR. PLUTA: Object to form. MR. PLUTA: Object to form. Do you see that? A. Yes. A. Yes. Do you see that? Do you see that? Do you see that? Do you see t				
A. From the slower memory, that's right. Q. So it's a penalty in time? A. That is right, and penalty in power consumption. Q. Great. So if you decrease the cache size, that will likely increase cache misses; right? MR. PLUTA: Object to form. BY THE WITNESS: A. What application do you have in mind? BY MR. TUMINARO: Q. I'm asking you in general. If you have a miss – a cache miss, right? A. Well, if your application is three lines, it will fit in a small cache. So you don't need miss, right? A. Wold if qualify your question with that. You asked me a general question. Q. Chay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. MR. PLUTA: Object to form.				•
10 Q. So it's a penalty in time? 11 A. That is right, and penalty in power 12 consumption. 13 Q. Great. So if you decrease the cache 14 size, that will likely increase eache misses; right? 15 MR. PLUTA: Object to form. 16 BY THE WITNESS: 17 A. What application do you have in mind? 18 BY MR. TUMINARO: 19 Q. I'm asking you in general. If you have a 20 smaller cache or fewer things in there, it is more 21 likely that you are going to have a miss a cache 22 miss, right? 23 A. Well, if your application is three lines, 24 it will fit in a small cache. So you don't need 25 what type of application youre talking about. 26 Q. All right. Since this case is about 27 decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU. right? 28 A. You didn't qualify your question with 29 that. You asked me a general question. 20 Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU. right? 29 A. You can write two lines of GPUs, and it still would be a very good size cache. So your alarge program, yes, you need a bigger cache. 20 BY MR. TUMINARO: 21 BY MR. TUMINARO: 22 BY MR. TUMINARO: 23 A. Correct. 24 (WHEREUPON, document marked as Exhibit 4 for identifications purposes. This is Us. Patent No. 723335. 4 Have you seen this document before? 4 A. Yes. 4 Correct. 20 And this is listed on your materials 21 (WHEREUPON, document marked as Page 37 22 Exhibit No. 5.) 23 A. Correct. 24 (WHEREUPON, document marked as Exhibit No. 5.) 25 A. Correct. 26 Q. And this is listed on your materials 27 Exhibit No. 5.) 28 YMR. TUMINARO: 28 YMR. TUMINARO: 39 Q. All right. Since this case is about and that type of application your question with that. You asked me a general question. 40 GPUs, left stalk about GPUs. 50 A. You didn't qualify your question with that. You asked me a general question. 51 A. You can write two lines of GPUs, and it still still out the page 4 Augustic and		• •		•
11 A. That is right, and penalty in power consumption. 22 Consumption. 33 Q. Great. So if you decrease the cache size, that will likely increase cache misses; right? 44 Size, that will likely increase cache misses; right? 55 MR. PLUTA: Object to form. 56 BY THE WITNESS: 57 A. What application do you have in mind? 58 BY MR. TUMINARO: 59 Q. I'm asking you in general. If you have a misser a cache misser, right? 50 Ilkely that you are going to have a miss a cache miss, right? 51 A. Well, if your application is three lines, it is more likely that you are going to have a miss a cache miss, right? 50 A. Well, if your application is three lines, it will fit in a small cache. So you don't need likely that you are going to have a miss a cache miss, right? 51 that. So you have to qualify your question is for what type of application you're talking about. 52 Q. All right. Since this case is about decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? 53 A. You can write two lines of GPUs, and it still most still not sufficient for me to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. 54 BY THE WITNESS: 55 A. A you can write two lines of GPUs, and it alarger program, yes, you need a bigger cache. 56 BY THE WITNESS: 57 A. A you can write two lines of GPUs, and it alarger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? 58 A. A larger program in a typical type GPU misses? 59 MR. PLUTA: Object to form. 50 A. Publication is still not sufficient for me to answer that. But let me help you out. If you want a larger program in a typical type GPU misses? 50 A. Publication is still not sufficient for me to answer that. But let me help you out. If you want and the publication is still not sufficient for me to answer that. But let me help you out. If you want and the add. The plurality of command processing engines, coupled to the arbiter, each operabl				WIK. TOWIIVAKO. Okay. Gleat.
12 Consumption. 13 Q. Great. So if you decrease the cache 13 BY MR. TUMINARO: Q. You have been handed what has been marked 15 MR. PLUTA: Object to form. 15 as Exhibit 4 for identifications purposes. This is U.S. Patent No. 7233335. Have you seen this document before? A. Yes. Q. This is the Morton patent, correct? A. Correct. Q. And this is listed on your materials considered, correct? A. Correct. Q. And this is listed on your materials considered, correct? A. Correct. Q. And this is listed on your materials considered, correct? A. Correct. Q. Millinght. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to a larger program, yes, you need a bigger cache. 14				(WHEDELIDON, document marked as
Q. Great, So if you decrease the cache size, that will likely increase cache misses; right? MR. PLUTA: Object to form. MR. PLUTA: Object to				
size, that will likely increase cache misses; right? MR. PLUTA: Object to form. A. What application do you have in mind? BY THE WITNESS: A. What application do you have in mind? BY MR. TUMINARO: Q. I'm asking you in general. If you have a making out in general. If you have a cache miss, right? A. Well, if your application is three lines, it is more likely that you are going to have a miss — a cache miss, right? A. Well, if your application is three lines, it will fit in a small cache. So you don't need A. Well, if your application is three lines, it will fit in a small cache. So you don't need A. Well, if your application is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. MR.		*		· · · · · · · · · · · · · · · · · · ·
15 MR. PLUTA: Object to form. BY THE WITNESS: 16 U.S. Patent No. 7233335. 17 A. What application do you have in mind? 18 BY MR. TUMINARO: 19 Q. I'm asking you in general. If you have a smaller cache or fewer things in there, it is more 20 smaller cache or fewer things in there, it is more 21 likely that you are going to have a miss a cache miss, right? 22 miss, right? 23 A. Well, if your application is three lines, it will fit in a small cache. So you don't need Page 35 1 that. So you have to qualify your question is for what type of application you're talking about. 24 GPUs, let's talk about GPUs. 25 A. You didn't qualify your question with that. You asked me a general question. 26 Q. Okay, How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? 16 MR. PLUTA: Object to form. 17 MR. PLUTA: Object to form to a larger program, yes, you need a bigger cache. 18 Q. Al arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? 18 MR. PLUTA: Object to form. 29 MR. PLUTA: Object to form. 30 Q. All arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? 20 MR. PLUTA: Object to form. 21 BY MR. TUMINARO: 22 MR. PLUTA: Object to form. 23 GPUs decrease the cache size will increase cache misses? 24 MR. PLUTA: Object to form. 25 MR. PLUTA: Object to form. 26 Do you see that? 27 A. Yes. 28 Do you see that? 29 Do you see that? 20 Do you see that? 20 Do you see that? 21 Do you see that? 22 Do you see that? 23 A. Yes.				
16 BY THE WITNESS: A. What application do you have in mind? BY MR. TUMINARO: Q. I'm asking you in general. If you have a smaller cache or fewer things in there, it is more likely that you are going to have a miss — a cache miss, right? A. Well, if your application is three lines, it will fit in a small cache. So you don't need Page 35 that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay, How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to a larger program, yes, you need a bigger cache. BY MR. PLUTA: Object to form. BY MR. TUMINARO: Q. Al arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY MR. TUMINARO: Q. Al arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY MR. TUMINARO: Q. Al arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY MR. TUMINARO: MR. PLUTA: Object to form. MR. PLUTA		·		-
A. What application do you have in mind? BYMR. TUMINARO: Q. I'm asking you in general. If you have a smaller cache or fewer things in there, it is more likely that you are going to have a miss a cache miss, right? A. Well, if your application is three lines, it will fit in a small cache. So you don't need Page 35 that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about 4 GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. All arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY MR. TUMINARO: Q. All arger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY MR. TUMINARO: Q. It's listed on your materials considered? A. Correct. (WHEREUPON, document marked as Exhibit No. 5.) BY MR. TUMINARO: A. Correct. Q. Sir, you have been handed what has been marked as Exhibit 5 for identification purposes. This is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. A. Correct. Q. It's listed on your materials considered? A. Correct. Q. If you would urn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Do you see that? A. Yes.		· ·		
18 BY MR. TUMINARO: Q. I'm asking you in general. If you have a smaller cache or fewer things in there, it is more likely that you are going to have a miss a cache miss, right? 22 A. Well, if your application is three lines, it will fit in a small cache. So you don't need 23 A. Well, if your application is three lines, it will fit in a small cache. So you don't need 24 by a considered, correct? 25 A. Correct. CWHEREUPON, document marked as 26 Page 37 27 by Mart type of application you're talking about. 28 GPUs, let's talk about GPUs. 29 All right. Since this case is about 40 GPUs, let's talk about GPUs. 40 A. Correct. CWHEREUPON, document marked as 41 Exhibit No. 5.) 42 BY MR. TUMINARO: 43 Q. All right. Since this case is about 44 GPUs, let's talk about GPUs. 45 A. You didn't qualify your question with that. You asked me a general question. 46 GPUs, let's talk about GPUs. 47 A. Correct. 48 Correct. CWHEREUPON, document marked as 49 BY MR. TUMINARO: 40 Correct. CWHEREUPON, document marked as 40 Q. Sir, you have been handed what has been marked as Exhibit No. 5.) 40 BY MR. PLUTA: Object to form. 40 Correct. 41 A. Correct. 42 Correct. 43 Correct. 44 Correct. 45 Correct. 46 Correct. 46 Correct. 47 Correct. 48 Correct. 49 Correct. 40 Correct. 40 Correct. 40 Correct. 41 Correct. 41 Correct. 42 Correct. 43 Correct. 44 Correct. 45 Correct. 46 Correct. 47 A. Correct. 48 Correct. 49 It is is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. 49 Correct. 40 Correct. 40 Correct. 41 Correct. 41 Correct. 42 Correct. 43 Correct. 44 Correct. 44 Correct. 45 Correct. 46 Correct. 46 Correct. 47 A. Correct. 48 Correct. 49 It is is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. 49 Correct. 40 Correct. 40 Correct. 41 Correct. 41 Correct. 42 Correct. 43 Correct. 44 Correct. 44 Correct. 44 Correct. 45 Correct. 46 Correct. 46 Correct. 47 A. Correct. 48 Correct. 49 It is is ted on your materials considered? 40 Correct. 40 Correct. 41 Correct. 41 Correct.				
Q. I'm asking you in general. If you have a smaller cache or fewer things in there, it is more likely that you are going to have a miss a cache miss, right? A. Well, if your application is three lines, it will fit in a small cache. So you don't need Page 35 that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. Page 35 Page 37 Exhibit No. 5.) BY MR. TUMINARO: D. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. A. Correct. Q. It's listed on your materials considered? (WHEREUPON, document marked as Page 37 Exhibit No. 5.) BY MR. TUMINARO: D. Okay. How about in GPUs? If you application purposes. This is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. A. Correct. Q. It's listed on your materials considered? A. Correct. Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes.		**		
20 smaller cache or fewer things in there, it is more likely that you are going to have a miss a cache miss, right? 21				
21 likely that you are going to have a miss a cache miss, right? 22 23 A. Well, if your application is three lines, 24 it will fit in a small cache. So you don't need 24 (WHEREUPON, document marked as Page 35 Page 37 1 Exhibit No. 5.) Page 37 Pa				-
22 miss, right? A. Well, if your application is three lines, it will fit in a small cache. So you don't need Page 35 that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: Q. Al arger program in a typical type GPU that you vould use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY MR. TUMINARO: Q. Okay. How about in GPUs, and it alarger program, yes, you need a bigger cache. MR. PLUTA: Object to form to decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form to decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form to decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form to decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form. MR. PLUTA: Object to form to decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form. MR. PLUTA: Object to form. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." MR. PLUTA: Object to form. Do you see that? A. Yes.			20	
23 A. Well, if your application is three lines, 24 it will fit in a small cache. So you don't need Page 35 that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form. MR. PLUTA: Object to form et to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. MR. PLUTA: Object to form to MA. Yes. Q. Alarger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Ob				·
24 it will fit in a small cache. So you don't need Page 35 that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. Page 37 Exhibit No. 5.) BY MR. TUMINARO: Q. Sir, you have been handed what has been marked as Exhibit 5 for identification purposes. This is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. A. Correct. Q. If's listed on your materials considered? A. Correct. Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? MR. PLUTA: Object to form. BY THE WITNESS: A. Yes.	22		22	considered, correct?
that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Dy MR. PLUTA: Object to form to answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. Page 37 Exhibit No. 5.) BY MR. TUMINARO: A. Correct. A. Correct. Q. If you would urn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes.			23	
that. So you have to qualify your question is for what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it alarger program, yes, you need a bigger cache. BY MR. TUMINARO: Description Exhibit No. 5.) BY MR. TUMINARO: Q. Sir, you have been handed what has been marked as Exhibit 5 for identification purposes. This is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. A. Correct. Q. It's listed on your materials considered? A. Correct. Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. A. Yes. Q. A larger program in a typical type GPU Hat you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? MR. PLUTA: Object to form. MR. PLUTA: Object to form. Do you see that? A. Yes. Do you see that? Do you see that? Do you see that?	24	it will fit in a small cache. So you don't need	24	(WHEREUPON, document marked as
what type of application you're talking about. Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it alarger program, yes, you need a bigger cache. PY MR. TUMINARO: Q. Sir, you have been handed what has been marked as Exhibit 5 for identification purposes. This is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. A. Correct. Q. It's listed on your materials considered? A. Correct. Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. Alarger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. MR. PLUTA: Object to form. A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes. Do you see that? A. Yes.		Page 35		Page 37
Q. All right. Since this case is about GPUs, let's talk about GPUs. A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. A. You asked me a general question. A. Correct. MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it qualification is still not sufficient for me to a larger program, yes, you need a bigger cache. BY MR. TUMINARO: A. Yes. Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. A. You can write two lines of GPUs, and it a larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes. A. Yes. Do you see that? A. Yes. Do you see that? A. Yes.	1	that. So you have to qualify your question is for	1	Exhibit No. 5.)
4 GPUs, let's talk about GPUs. 5 A. You didn't qualify your question with 6 that. You asked me a general question. 7 Q. Okay. How about in GPUs? If you 8 decrease the cache size, it will likely increase the 9 likelihood of a cache miss in a GPU, right? 9 A. Correct. 10 MR. PLUTA: Object to form. 11 BY THE WITNESS: 12 A. You can write two lines of GPUs, and it 13 still would be a very good size cache. So your 14 qualification is still not sufficient for me to 15 a larger program, yes, you need a bigger cache. 16 BY MR. TUMINARO: 17 A. Yes. 18 Q. A larger program in a typical type GPU 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 24 marked as Exhibit 5 for identification purposes. 25 This is U.S. Patent No. 7742053. This is the '053 26 patent that we have been talking about. 26 A. Correct. 28 Q. It's listed on your materials considered? 39 A. Correct. 30 Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, 31 Sir? 32 A. Yeah. 33 Q. The last element in claim No. 5 is a plurality of command processing engines. 30 Po you see that? 31 A. Yes. 32 Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 34 Do you see that? 35 Do you see that? 36 Do you see that? 37 Do you see that? 38 PTHE WITNESS: 38 A. Yes.	2	what type of application you're talking about.	2	
A. You didn't qualify your question with that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it qualification is still not sufficient for me to a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. 5 This is U.S. Patent No. 7742053. This is the '053 patent that we have been talking about. 7 A. Correct. 8 Q. It's listed on your materials considered? 9 A. Correct. 9 Lif you would turn with me to the last column, column A in claim No. 5. Are you there, sir? 11 Still would be a very good size cache. So your 13 A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." MR. PLUTA: Object to form. 20 MR. PLUTA: Object to form. 21 Do you see that? A. Yes.	3	Q. All right. Since this case is about	3	Q. Sir, you have been handed what has been
that. You asked me a general question. Q. Okay. How about in GPUs? If you decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it would be a very good size cache. So your qualification is still not sufficient for me to a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." MR. PLUTA: Object to form. MR. PLUTA: Object to form. A. Correct. A. Yea would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. A. Yesh. Do you see that? A. Yes.	4	GPUs, let's talk about GPUs.	4	marked as Exhibit 5 for identification purposes.
7Q. Okay. How about in GPUs? If you7A. Correct.8decrease the cache size, it will likely increase the9Q. It's listed on your materials considered?9likelihood of a cache miss in a GPU, right?9A. Correct.10MR. PLUTA: Object to form.10Q. If you would turn with me to the last11BY THE WITNESS:11column, column A in claim No. 5. Are you there,12A. You can write two lines of GPUs, and it12sir?13still would be a very good size cache. So your13A. Yeah.14qualification is still not sufficient for me to14Q. The last element in claim No. 5 is a15answer that. But let me help you out. If you want15plurality of command processing engines.16a larger program, yes, you need a bigger cache.16Do you see that?17A. Yes.Q. And I'll read it for the record. "A19that you would use in a commercial implementation,19plurality of command processing engines, coupled to20decreasing the cache size will increase cache20the arbiter, each operable to receive and process21misses?21Do you see that?22MR. PLUTA: Object to form.22Do you see that?23BY THE WITNESS:23A. Yes.	5	A. You didn't qualify your question with	5	This is U.S. Patent No. 7742053. This is the '053
decrease the cache size, it will likely increase the likelihood of a cache miss in a GPU, right? MR. PLUTA: Object to form. BY THE WITNESS: A. You can write two lines of GPUs, and it qualification is still not sufficient for me to a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. A larger program in a typical type GPU and that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY THE WITNESS: BY THE WITNESS: A. Correct. Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, sir? A. Yeah. Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes.	6	that. You asked me a general question.	6	patent that we have been talking about.
9 likelihood of a cache miss in a GPU, right? 10 MR. PLUTA: Object to form. 11 BY THE WITNESS: 12 A. You can write two lines of GPUs, and it 13 still would be a very good size cache. So your 14 qualification is still not sufficient for me to 15 answer that. But let me help you out. If you want 16 a larger program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 A. Correct. 29 A. Correct. 20 G. If you would turn with me to the last column, column A in claim No. 5. Are you there, 31 Sir? 32 A. Yeah. 34 Yeah. 45 Q. The last element in claim No. 5 is a plurality of command processing engines. 46 Do you see that? 47 A. Yes. 48 Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 49 Do you see that? 40 Do you see that? 41 Do you see that? 42 Do you see that? 43 A. Yes.	7	Q. Okay. How about in GPUs? If you	7	A. Correct.
MR. PLUTA: Object to form. 10 Q. If you would turn with me to the last column, column A in claim No. 5. Are you there, 12 A. You can write two lines of GPUs, and it 13 still would be a very good size cache. So your 14 qualification is still not sufficient for me to 15 answer that. But let me help you out. If you want 16 a larger program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 BY THE WITNESS: 24 Do you see that? 25 Do you see that? 26 Do you see that? 27 Do you see that? 28 Do you see that? 29 Do you see that? 20 A. Yes.	8	decrease the cache size, it will likely increase the	8	Q. It's listed on your materials considered?
11 BY THE WITNESS: 12 A. You can write two lines of GPUs, and it 13 still would be a very good size cache. So your 14 qualification is still not sufficient for me to 15 answer that. But let me help you out. If you want 16 a larger program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 BY THE WITNESS: 24 Column, column A in claim No. 5. Are you there, 25 sir? 26 A. Yeah. 27 Q. The last element in claim No. 5 is a plurality of command processing engines. 26 Do you see that? 27 A. Yes. 28 Do you see that? 29 Do you see that? 20 the arbiter, each operable to receive and process the command thread." 29 Do you see that? 20 Do you see that? 20 A. Yes.	9	likelihood of a cache miss in a GPU, right?	9	A. Correct.
A. You can write two lines of GPUs, and it still would be a very good size cache. So your qualification is still not sufficient for me to answer that. But let me help you out. If you want la larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. The last element in claim No. 5 is a plurality of command processing engines. Do you see that? A. Yes. Ryes. Q. A larger program in a typical type GPU la Q. And I'll read it for the record. "A plurality of command processing engines, coupled to decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. Ryeah. A. Yeah. Do you see that? A. Yes. Do you see that? A. Yes. Do you see that? A. Yes.	10	MR. PLUTA: Object to form.	10	Q. If you would turn with me to the last
still would be a very good size cache. So your qualification is still not sufficient for me to 14 Q. The last element in claim No. 5 is a 15 answer that. But let me help you out. If you want 16 a larger program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 BY THE WITNESS: 24 A. Yes. 25 A. Yes. 26 A. Yes. 27 Do you see that? 28 Do you see that? 29 Do you see that? 20 the arbiter, each operable to receive and process the command thread." 29 Do you see that?	11	BY THE WITNESS:	11	column, column A in claim No. 5. Are you there,
14qualification is still not sufficient for me to14Q. The last element in claim No. 5 is a15answer that. But let me help you out. If you want15plurality of command processing engines.16a larger program, yes, you need a bigger cache.16Do you see that?17BY MR. TUMINARO:17A. Yes.18Q. A larger program in a typical type GPU18Q. And I'll read it for the record. "A19that you would use in a commercial implementation,19plurality of command processing engines, coupled to20decreasing the cache size will increase cache20the arbiter, each operable to receive and process21misses?21the command thread."22MR. PLUTA: Object to form.22Do you see that?23BY THE WITNESS:23A. Yes.	12	A. You can write two lines of GPUs, and it	12	sir?
answer that. But let me help you out. If you want a larger program, yes, you need a bigger cache. BY MR. TUMINARO: Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. By MR. Tuminary of command processing engines. A. Yes. Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes.	13	still would be a very good size cache. So your	13	A. Yeah.
16 a larger program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 BY THE WITNESS: 24 Do you see that? 25 Do you see that? 26 Do you see that? 27 A. Yes. 28 Po you see that? 29 Do you see that? 20 the arbiter, each operable to receive and process the command thread." 29 Do you see that? 20 A. Yes.	14	qualification is still not sufficient for me to	14	Q. The last element in claim No. 5 is a
16 a larger program, yes, you need a bigger cache. 17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 BY THE WITNESS: 24 Do you see that? 25 Do you see that? 26 Do you see that? 27 A. Yes. 28 Program in a typical type GPU 29 Q. And I'll read it for the record. "A 29 plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 29 Do you see that? 20 And I'll read it for the record. "A 20 plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 20 And I'll read it for the record. "A 20 plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 20 And I'll read it for the record. "A 20 plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 21 And I'll read it for the record. "A 22 And I'll read it for the record. "A 23 And I'll read it for the record. "A 24 Plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." 25 And I'll read it for the record. "A 26 Plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread."	15	answer that. But let me help you out. If you want	15	plurality of command processing engines.
17 BY MR. TUMINARO: 18 Q. A larger program in a typical type GPU 19 that you would use in a commercial implementation, 20 decreasing the cache size will increase cache 21 misses? 22 MR. PLUTA: Object to form. 23 BY THE WITNESS: 21 A. Yes. 24 A. Yes. 25 A. Yes.	16		16	
Q. A larger program in a typical type GPU that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form. BY THE WITNESS: Q. And I'll read it for the record. "A plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Do you see that? A. Yes.				-
that you would use in a commercial implementation, decreasing the cache size will increase cache misses? MR. PLUTA: Object to form.	18		18	Q. And I'll read it for the record. "A
20decreasing the cache size will increase cache20the arbiter, each operable to receive and process21misses?21the command thread."22MR. PLUTA: Object to form.22Do you see that?23BY THE WITNESS:23A. Yes.				
21misses?21the command thread."22MR. PLUTA: Object to form.22Do you see that?23BY THE WITNESS:23A. Yes.				
22 MR. PLUTA: Object to form. 22 Do you see that? 23 BY THE WITNESS: 23 A. Yes.				-
23 BY THE WITNESS: 23 A. Yes.				
				-
A. Yes. You asked me questions that are 24 Q. And in the previous element, it cites an	23	DI THE WITNESS.		A. 168.

Page 38 Page 40 1 arbiter. The time is 10:09 a.m. 1 2 2 BY MR. TUMINARO: Do you see that? 3 A. Yes. 3 Q. If you would turn -- Welcome back, 4 Q. I'll read that one. "An arbiter, coupled 4 Dr. Bagherzadeh. Did you talk with counsel about 5 5 to be at least one memory device, operable to select the substance of your testimony during the break? 6 a command thread from either of the plurality of 6 pixel command threads and the plurality of vertex 7 7 Q. If you'd turn with me to Exhibit 1, your 8 command threads." 8 declaration again. In particular, I would like to 9 Do you see that? 9 go to Page 72. I'm sorry. Paragraph 72, this is 10 10 referring to the Lindholm reference, is that right, A. Yes. 11 Q. So looking at the plurality of command 11 starting on Page 20? 12 processing engines, each of those has to be able to 12 A. Correct. 13 process a vertex command thread or a pixel command 13 14 thread, right? 14 (WHEREUPON, document marked as 15 15 MR. PLUTA: Object to form. Exhibit No. 6.) 16 THE WITNESS: You are asking a yes or no 16 BY MR. TUMINARO: 17 question? 17 Q. You have been handed what's been marked 18 18 BY MR. TUMINARO: as Exhibit 6 for identification purposes. This is 19 Q. Yes. 19 U.S. Patent No. 7015913. 20 A. No. 20 A. Yes. 21 It does not? 21 Q. This is the Lindholm patent? Q. 22 No. 22 A. A. Correct. 23 23 When it says in claim 5, "each operable Q. This is listed on your materials 24 to receive and process the command thread," that's 24 considered. Page 39 Page 41 not referring to vertex command threads or pixel 1 1 A. Correct. 2 command threads? 2 Q. Now, Paragraph 72 of your declaration 3 3 A. It doesn't say that. It just says says and I'll read it for the record: "Because 4 receive and process. I don't get the --4 thread control buffer 420 stores information for 5 Q. It doesn't say "the command thread"? 5 both pixel and vertex threads, they are the claimed 6 A. Yeah, but it doesn't say that it should 6 memory device." 7 be able to do both. I don't see that here. 7 Do you see that? 8 Q. What do you understand the command thread 8 A. I see that. 9 9 to be then? Q. So, Paragraph 72 of your declaration is 10 1.0 A. It's -- it doesn't say what you're referring to -- just to be clear, the thread 11 saying. 11 controlled buffer 420, which is illustrated, for Q. You don't understand it to be a pixel 12 12 example, in figure 4 of Lindholm; is that right? 13 command thread or a vertex command thread? 13 A. Yes. 14 14 A. Not to do both. There will be some that Q. If you will look with me back to 15 would do pixel commands. The other ones would be to 15 Exhibit 5. 16 thread commands. 16 A. Yes. 17 Counsel, do you want to take a break? 17 Q. In claim 1, which appears on column 7, 18 MR. TUMINARO: Sure. Now is a good time 18 the memory device that's claimed in claim 1 reads, 19 19 and I'll read it for the record: "At least one to take a break. 20 THE VIDEOGRAPHER: We're going off 20 memory device comprising a first portion operative 21 record. The time is 9:55 a.m. 21 to store a plurality of pixel command threads and a 22 22 (Whereupon, a break in the second portion operative to store a plurality of 23 proceedings was taken.) 23 vertex command threads." 24 THE VIDEOGRAPHER: We're back on record. 24 Do you see that?

Page 42 Page 44 1 A. Yes. BY THE WITNESS: 1 2 2 Claim 1, the thing that's required to be A. It says to store a several pixel command 3 3 stored are the pixel command threads, command threads and several vertex command threads. threads -- strike that. Let me start over. It was 4 BY MR. TUMINARO: 5 5 all mixed up. Q. Okay. Great. We agree that it's pixel 6 In claim 1, the thing that is stored 6 command threads and vertex command threads, right? 7 7 by the memory device are the pixel command threads These are instructions for the threads, 8 and the vertex command threads; is that right? 8 yes. 9 9 MR. PLUTA: Object to form. Q. In Paragraph 72 of your declaration, what 10 BY THE WITNESS: 10 you say that is stored is "information" for both 11 A. You just repeated that claim, right? It 11 pixels and vertex threads; is that right? 12 talks about -- is that what you did? 12 A. You're reading my statement. 13 13 BY MR. TUMINARO: Q. I just want to be clear that you're saying information about the pixels and vertices 14 Q. What I'm trying to get at is the memory 14 15 device stores a plurality of pixel command threads 15 satisfies the claimed vertex command thread and 16 and a plurality of vertex command threads. 16 pixel command thread? 17 MR. PLUTA: Object to form, if that's a 17 A. I'm not saying that. 18 18 question. What are you saying then? 19 BY THE WITNESS: 19 A. It's a combination of TCB plus the 20 A. It says a memory device comprising of a 20 instruction cache that satisfies that particular 21 first portion operative to store a plurality of 21 element of claim 1. 22 pixel command threads and a second portion operative 22 Q. Well, in Paragraph 72 of your 23 to store a plurality of vertex command threads. I 23 declaration, you're saying that the TCB, as you put 2.4 24 agree with what it says. it, stores information, right? Page 43 Page 45 1 BY MR. TUMINARO: 1 A. Yes. 2 Q. I'm not trying to trick you. There are 2 Q. The claim requires command threads, 3 two things that are stored in the memory device, 3 right? 4 right? 4 That's what it says, yes. 5 MR. PLUTA: Object to form. 5 Q. I just want to be clear. So you're 6 BY MR. TUMINARO: 6 saying that the information stored by the TCB 7 7 Q. Let me try again. There are two types satisfies the claimed command threads? 8 that are stored by the memory device in claim 1; is 8 MR. PLUTA: Object to form. 9 9 BY THE WITNESS: that right? 10 10 MR. PLUTA: Object to form. A. I did not say that. 11 11 BY THE WITNESS: BY MR. TUMINARO: 12 12 Q. That's not what you're saying in A. You're saying at most two or at least two 13 or exactly two? 13 Paragraph 72? 14 14 BY MR. TUMINARO: A. All I'm saying is -- I think you're 15 15 mixing information with instructions. Q. At least two. 16 MR. PLUTA: Still objection. 16 Q. That's what I'm trying to understand. 17 BY THE WITNESS: 17 I'm trying to understand exactly what it is that 18 18 A. I would say it says two here right now. you're mapping in Lindholm that corresponds to the 19 I would not know if there was more or less -- I mean 19 claimed vertex command threads and pixel command 20 20 more. 21 BY MR. TUMINARO: 21 A. As I think I just alluded to just a few 22 22 Q. So what is stored, what does the claim minutes ago, it's a combination of TCB plus the 23 23 require that the memory device stores? instruction cache. 24 MR. PLUTA: Object to form. 24 Q. So what is the thing that is stored in

Page 46 Page 48 the TCB and instruction cache that you're saying is 1 Lindholm does not explicitly disclose thread -- a 2 2 the claimed command threads? thread control buffer that has a first portion for 3 3 A. It is the instructions plus information. storing pixel command threads and a second portion 4 Q. Just to be clear -- I guess what I'm not 4 for storing vertex command threads? 5 5 MR. PLUTA: Object to form. clear about is what is the information that you're 6 talking about? 6 BY THE WITNESS: A. Okay. 7 7 A. Explicitly? I say it, but a POSITA would 8 MR. PLUTA: Object to form. 8 have figured it out from the block diagram and the 9 BY MR. TUMINARO: 9 knowledge that it is --10 Q. All right. Let me clarify. No, I 10 BY MR. TUMINARO: 11 appreciate his objection. So let me clarify. 11 Q. Okay --12 In Paragraph 72 of your declaration, 12 A. -- a combination. 13 what information is it that you're talking about? 13 Q. -- I'm not talking about that for a 14 A. Let me refer you to my report. I wish I 14 second. I'll get to that. I'll get to that. I 15 15 could do a search. promise you. Just for now you would agree that 16 16 there is no explicit disclosure in Lindholm about a Referring to my claim chart on 17 Page 25 of my report, it's just an example of this. 17 first portion that stores pixel command threads and 18 18 It says TCB or thread control buffer 420 includes a second portion that stores vertex command threads? 19 19 storage resources to retain thread state data for a A. So taking the first part of that 20 subset of predetermined number of threads and that's 20 sentence, you are repeating it correctly, but there 21 the data. It's the state data because you have to 21 is more to that. 22 track this information while instructions are alive, 22 So you would agree with me -- strike Q. 23 23 and once they retire, you let go of that state. that. 24 Q. Just to be clear, the thread control 24 And that storing of a first portion Page 49 Page 47 buffer 420 stores this state data that you just operative to store a plurality of pixel command 2 referred to? 2 threads and a second portion operative to store a 3 3 A. At least. plurality of vertex command threads, that's a 4 Q. At least the state data. And that state 4 requirement of claim 1 of the '053 patent, right? 5 data corresponds to the information that you refer 5 You could look at the '053 patent if 6 to in Paragraph 72 of your declaration? 6 you need to. 7 7 A. Some of the information is state data. A. You repeated claim 1, the first element 8 right. 8 of that claim 1, right? 9 9 Q. If you turn with me to Paragraph 73 of O. Right. 10 your declaration. 10 A. Your question is that is a requirement? 11 11 A. Okay. Q. Yes. 12 Q. And the first word is "while" in that 12 A. It's a requirement and Lindholm satisfies 13 paragraph, right? 13 it. 14 14 A. Yes. Q. You would agree with me that Lindholm 15 Q. After that, it reads: "The thread 15 doesn't explicitly disclose it, right? 16 control buffer 42 does not explicitly disclose a 16 MR. PLUTA: Object to form. 17 first portion for storing pixel command threads or a 17 BY THE WITNESS: 18 second portion for storing vertex command threads." 18 A. I think it says in my report, but it's my 19 19 Do you see that? opinion that it would have been obvious to one of 20 20 A. I think you read it correctly. the ordinary skill. Because if you take the 21 Q. Is that a true statement? It's an 21 sentence out of the context, just the first part, 22 22 accurate statement? it's not really my position. 23 23 A. Everything I have here is true. BY MR. TUMINARO: 24 Okay. So you would agree with me that 24 Q. I appreciate that. I just want to be

Page 50 Page 52 1 right? clear on what your position is. I'll get to obviousness. I promise. You are not saying that 2 2 A. It's Lindholm in view of AAPA, that's 3 3 Lindholm anticipates claim 1, right? right. 4 4 Q. And the AAPA that you're relying on, you A. I don't think we said that it anticipates 5 5 talk about that in Paragraph 100, right? claim 1. 6 6 A. Correct. Q. Okay. Okay. 7 7 Q. You mention here in Paragraph 100, it A. I don't think we did, but I can check for 8 says, "the '053 patent discloses that buffer 104 8 you. In view of AAPA, so we're not saying 9 anticipates, right. I'm not saying it's 9 stores ALU resource command threads while buffer 106 stores texture fetch resource command threads." 10 10 anticipating that. 11 11 Do you see that? Q. I just want to be clear. You agree 12 Lindholm does not anticipate claim 1? 12 A. Correct. 13 13 Q. Okay. So this background material talks A. On it's own, correct. about ALU threads and texture threads, right? 14 Q. Okay. All right. But you think that 14 15 15 MR. PLUTA: Object to form. it's obvious, right? BY THE WITNESS: 16 16 A. Yes. 17 Q. Okay. I told you I would get to 17 A. It talks about instructions belonging to 18 obviousness. That's what I'm trying to get to. 18 the -- okay. 19 BY MR. TUMINARO: MR. PLUTA: Object to form. 19 20 20 Q. Let me clarify. There's a first buffer, BY MR. TUMINARO: 21 21 buffer 104 for storing ALU threads, resource command Q. And the reason -- if you look at 22 22 threads, and a second buffer, 106, for storing Paragraph 73, it says about -- after that first 23 23 texture fetch resource command threads, right? subordinate clause that I read, it says: "It is my 24 opinion." 24 A. Yes. Page 51 Page 53 1 1 Do you see that in Paragraph 73? Q. Okay. If you go back to claim 1 of the 2 A. Yes. 2 '053 patent, claim 1 is not about -- it doesn't say 3 Q. I'll read it for the record just so it's 3 anything about ALU command threads or texture 4 clear. "It is my opinion that it would have been 4 command threads. It talks about pixel command 5 obvious to one of ordinary skill in the art to 5 threads and vertex command threads, correct? 6 modify Lindholm's multi-threaded system to store 6 A. That's what it says. But I can qualify 7 7 different types of command threads in separate that if you are trying to tie it into this 8 portions of memory because it is well known that 8 paragraph. 9 9 pixel threads and vertex threads are necessarily Q. I guess in paragraph -- okay. In 10 10 different types of data." Paragraph 101, you say: "In graphics processing, it 11 11 Do you see that? is well known that vertex command threads belong in 12 12 A. Yes. the category of ALU resource division because of the 13 Q. You stand by that statement? 13 nature of vertex operations." 14 14 MR. PLUTA: I'm going to object, Counsel. A. Correct. 15 15 This is outside the scope of the instituted grounds. Q. You provide no cite for that, correct? 16 BY MR. TUMINARO: 16 A. It says well known, in my experience, 17 17 teaching, research. Q. If you would turn with me to 18 Paragraph 100 of your declaration. 18 Q. You think it's a true statement? 19 19 A. It is. A. Yep. 20 20 In fact, pixel commands can also be ALU Q. And Paragraph 100, this is related to Q. 21 your position that this paragraph falls within the 21 operations; is that right? 22 22 section related to ground No. 2 where you're saying Sometimes. Predominantly, it is vertex. 23 23 that Lindholm and so-called admitted prior art Q. You didn't qualify it like that in 24 render obvious certain claims of '053; is that 24 Paragraph 101, did you?

Page 54 Page 56 1 A. It's obvious. 1 Multiplication could be one of them, but it is 2 2 Q. But you said -- so you would agree with usually not floating point. It's integer. me then that pixel command threads could also be ALU 3 BY MR. TUMINARO: operations, right? 4 Q. Multiplication is an ALU, not a texture 5 5 MR. PLUTA: Object to form. type of operation, right? 6 BY THE WITNESS: 6 MR. PLUTA: Object to form. 7 A. Predominantly, it's vertex. 7 BY THE WITNESS: 8 BY MR. TUMINARO: 8 A. It is multiplication. It is ALU. 9 Q. I don't think you answered my question. 9 BY MR. TUMINARO: 10 10 You would agree with me that pixel command threads Q. So, in fact, alpha blending is an example 11 can also be ALU operations? 11 of a pixel command that involves ALU operations? 12 MR. PLUTA: Object to form. 12 MR. PLUTA: Object to form. It is also 13 BY THE WITNESS: 13 outside the scope of the report. 14 A. They could, but it's not very common. 14 BY THE WITNESS: 15 BY MR. TUMINARO: 15 A. I think I have answered you to the best of my abilities. I explained to you that 16 Q. Alpha blending would be an example of 16 17 which pixels commands would be associated with an 17 predominantly it is vertex operations. I don't have 18 18 ALU type operation, right? anything more to add to that. 19 MR. PLUTA: Object to form, lack of 19 BY MR. TUMINARO: 20 foundation. 20 Q. In paragraph 101 you say that vertex 21 BY THE WITNESS: 21 command threads belong to the category of ALU 22 A. As an architect, when we evaluate 22 resource division, right? 23 23 systems, we look at dominance of certain functions A. That's what it says. 24 2.4 in terms of performance, power consumption, and so Q. But isn't it a fact that vertex commands Page 55 Page 57 1 on. A POSITA would have looked at this and would can also involve texture operations? 2 have said, yeah, vertex computations are taking a 2 A. Well, again, predominantly they're 3 lot of computations in terms of using the ALUs. So, 3 computation intensive operations. So they belong to 4 yeah. It would be the dominant one, 80, 70 percent, 4 the ALU sections. 5 and that's -- that's what you do, part of the 5 Q. But it is a fact that vertex commands can 6 6 design. involve texture operations, right? 7 7 BY MR. TUMINARO: MR. PLUTA: Object to form. 8 8 BY THE WITNESS: Q. None of that answered my question, 9 9 respectfully. A. It's a small percentage in terms of 10 10 A. I'm so sorry about that. overall computation. So if you want to take a 11 11 Q. Alpha blending would be an example in hundred to be the example, just using as an example, 12 12 which -- well, strike that. a hundred to be the amount of time, 80 percent 13 Alpha blend is a pixel type 13 perhaps would be -- 90 percent would be in the ALU 14 14 operation? portion, there will be a small portion for texture. 15 15 A. It is. But it's not the dominant BY MR. TUMINARO: 16 computation in terms of -- how you design the 16 Q. Just to be clear, the answer to my 17 17 architecture is you want to be able to address the question is yes? 18 18 most computation intensive functions. So, vertex MR. PLUTA: Object to form. 19 19 BY MR. TUMINARO: will dominant here. 20 20 Q. In alpha blending, the pixel command Q. Vertex commands can involve texture 21 involves multiplication, right? 21 operations? 22 22 MR. PLUTA: Object to form. A. I explained to you that it's 23 23 BY THE WITNESS: predominantly a computation of ALU. 24 A. It includes some sort of a computation. 24 Q. But it can be -- vertex commands can be

Page 58 Page 60 1 something, select one of two different things, you texture operations? 1 2 MR. PLUTA: Object to form, asked and 2 have to have the option to select either of those 3 answered. 3 two things, right? 4 BY THE WITNESS: 4 A. You could have a computation that only 5 5 A. I cannot add any more to what I just the vertex is available. I mean, it's not an --6 said. I mean, it is what it is. You can design it 6 it's not against whatever they say here. It could 7 the way you want. But it's predominantly a 7 be just one chute, if I could use the word chute, is 8 8 computation of the ALU. available, and you just pick from that one. 9 9 BY MR. TUMINARO: Q. Well, if I say to you do you want pizza 10 Q. The morphosys chip that you developed --10 or a sub for lunch, you select. You have to have 11 A. We're switching topics back to morphosys? 11 both of those options available at the same time, 12 Is that what you are asking me? 12 right? 13 Q. Yes. Could that system handle alpha 13 A. If the Subway is closed, I'm going to 14 blending? 14 have pizza. So until the subway opens. 15 15 MR. PLUTA: Object to form. Object to Q. In which case you haven't made a 16 16 the relevance. selection, right? 17 BY THE WITNESS: 17 A. Sure. I made a selection, but it was not 18 18 available. A. Yes. 19 BY MR. TUMINARO: 19 Q. Right but you didn't make that the 20 Q. If you will turn with me to the '053 20 selection, right? 21 patent and claim 1. 21 A. No. I had the option to going to the 22 A. Yes, sir. 22 Subway, but it was closed. It was not available. 23 23 Q. And the second element is an arbiter. Q. Is it your position if an option is not 24 And it reads, for the record, "an arbiter, coupled 24 available to you, you're still selecting that Page 59 Page 61 1 to be at least one memory device, operable to select 1 option? 2 a command thread from either of the plurality of 2 A. In what context are we talking about now? 3 3 The Subway or the computer architecture? pixel command threads and the plurality of vertex 4 command threads." I'll stop there. 4 Q. Let's talk about the context of the 5 5 Do you see that? arbiter. If the pixel is not available to the 6 A. I see that. 6 arbiter, would you say the arbiter is selecting a 7 7 Q. So the claimed arbiter has to select from vertex even though a pixel is not available to it? two types of things, right, vertex command threads 8 A. The job of the arbiter -- I have to 9 9 explain to you, Counsel, if you allow me to finish. and pixel command threads? 10 MR. PLUTA: Object to form. 10 If you are not going to allow me to finish, then I 11 11 BY THE WITNESS: won't be able to answer your question. 12 12 Q. Sir, I didn't say anything. A. The arbiter has to select a command 13 thread from either of the plurality of pixel command 13 A. The way you shaked your face, it was like 14 14 threads. Yeah, that's what it says. you were frustrated with me. 15 15 BY MR. TUMINARO: The job of the arbiter is to select 16 Q. In order to select either a pixel or a 16 between the available options. Is that clear what I 17 vertex, those options both must be available to the 17 said? And that's all it does. You gave an example 18 18 arbiter at the same time, right? of two for this particular claim. It could be 19 19 three, it could be four, and depending when it is MR. PLUTA: Object to form. BY THE WITNESS: 20 20 available. So it's very simple. No tricks. 21 A. What do you mean, those options have to 21 Q. In claim 1, the available options are 22 22 be available? I don't understand that. pixel command threads and vertex command threads? 23 23 A. If they're available, yes. BY MR. TUMINARO: 24 Q. Well, if you are going to select 24 Q. In claim 1, they are available, right?

	Page 62		Page 64
1	MR. PLUTA: Object to form.	1	BY MR. TUMINARO:
2	BY THE WITNESS:	2	Q. Well, it has to select?
3	A. Does it say that, that it has to be	3	A. It has to select. I agree with that.
4	available? I don't see that.	4	MR. PLUTA: Object to form.
5	BY MR. TUMINARO:	5	BY MR. TUMINARO:
6	Q. Well, it doesn't say if they are	6	Q. Does a FIFO select? Do you know what a
7	available, does it?	7	FIFO is?
8	A. Yeah. So, to be more general, it doesn't	8	A. First-in first-out.
9	say it's a necessary condition for the arbiter to	9	Q. First-in first-out buffer?
10	work.	10	A. Okay.
11	Q. I apologize.	11	Q. Does a FIFO select?
12	A. If you let me finish the thought, please.	12	A. You could design a FIFO that has a select
13	I'm trying to clarify what's on the claim. As you	13	line.
14	probably know better than I do, these claims	14	Q. But Does a FIFO by itself select?
15	sometimes are not very well written.	15	A. I mean, you're talking about a FIFO
16	But in this case it doesn't say the	16	without giving me any information about what you
17	requirement of having both available. It says it	17	want to do. So it's a very broad question. If I
18	has the option of choosing between the two. I agree	18	give you a FIFO description, and I say I want to
19	with that.	19	have an arbiter in front of it, you can design it.
20	Q. You said the job of the arbiter is to	20	Q. I'll tell you what. What is a FIFO?
21	select between available options?	21	A. First-in first-out.
22	A. Yeah.	22	O. What does it do?
23	Q. Right. You're talking about the claimed	23	A. The first-in first-out, basically that's
24	arbiter.	24	how it operates.
	Page 63		Page 65
1	A. That was a general comment, but it	1	Q. It stores the first thing that it gets
2	applies here.	2	and it outputs the first thing that's input is
3	Q. The claimed arbiter, the options are	3	the first thing that's output, is that what you're
4	pixel command threads and vertex command threads?	4	saying?
5	A. If they're available.	5	A. I think FIFO means that.
6	Q. If they're available is not a limitation	6	Q. So does a FIFO select between available
7	that appears in claim 1, right?	7	options?
8	A. It doesn't say that.	8	A. You can design it to do that.
9	Q. Okay.	9	Q. How would you design it to do that?
10	A. But it could be interpreted by whoever is	10	A. You put an arbiter in it.
11	looking at this.	11	Q. So a FIFO is not an arbiter itself?
12	Q. But it's not a limitation that's in the	12	A. No. It depends on what you want to do.
13	claim?	13	If you want it to have an arbitration, you could. I
14	MR. PLUTA: Object to form.	14	could design a FIFO with an arbiter.
15	BY THE WITNESS:	15	Q. You would have to put an arbiter in front
16	A. Right.	16	of the FIFO for the FIFO arbitrate?
17	BY MR. TUMINARO:	17	A. You have to put the functionality in the
18	Q. So an arbiter, the claimed arbiter, has	18	FIFO. If you need that, you put it in, and you call
19	to decide or resolve something?	19	that a FIFO. That will be my FIFO because I need
20	MR. PLUTA: Object to form.	20	that functionality. There is no set well, you
21	BY THE WITNESS:	21	could look up a dictionary and come up and say FIFO
22	A. I'm not sure if I understand that	22	means this; but if I want to design it, I can add
23	question.	23	anything I want to it and it will be a FIFO. I have
24		24	done that.

	Page 66		Page 68
1	Q. Do you know what a bus is in a computer	1	write information.
2	context?	2	A. In the context of, sorry, computer
3	A. Yes, I do.	3	architecture, yes.
4	Q. A bus provides data?	4	Q. In the context of computer architecture?
5	A. It does.	5	A. In general, yes.
6	Q. It receives data?	6	Q. Would that be true also in a GPU?
7	A. Not only data.	7	MR. PLUTA: Counsel, I'm going to object.
8	Q. What else does it receive?	8	I gave you a little leeway as the witness said. Can
9	A. Other things.	9	you tie this back to the scope of his report and
10	Q. Is a bus an arbiter?	10	particular information in his report? If not, I'm
11	A. It could be.	11	going to object to the relevance at this point.
12	Q. How?	12	MR. TUMINARO: Counsel, I'll draw your
13	A. Put an arbiter in it as part of the bus.	13	attention to the scheduling order, and it reminds
14	Q. So the arbiter is something different	14	everyone about the testimonial guidelines, and the
15	than the bus?	15	testimonial guidelines are explicit that objections
16	A. It could be integrated with the bus.	16	should be limited to a single word or term.
17	Q. What's a register?	17	Examples of objections that would be properly stated
18	A. It's memory.	18	are objection, form; objection, hearsay; objection,
19	Q. Memory. What's a register do?	19	relevance; and objection, foundation.
20	A. Store information.	20	So I'm going to ask you to refrain
21	Q. So it could receive information?	21	from speaking objections from now on.
22	A. It has to if it wants to store	22	
23	information.	23	
24	Q. It could provide information?	24	
	Page 67		Page 69
1	A. It can read it and write it.	1	MR. PLUTA: Then I'll restate my
2	Q. Is a register an arbiter?	2	objection. I object to the relevance of this line
3	A. Tough question.	3	of questioning.
4	Q. You are the expert.	4	BY MR. TUMINARO:
5	A. You are asking a very abstract question.	5	Q. If you would turn with me to in your
6	A register is part of the arbiter.	6	declaration to Page 10. And there's a figure at the
7	Q. It is part of an arbiter?	7	top of Page 10, the figure from your lectures.
8	A. If you design an arbiter without a	8	A. Yes.
9	register, you have a design that may not work.	9	Q. On the right-hand block that says CPU and
10	Q. So it's a portion of an arbiter. What	10	on the left-hand block, there's a block inside with
11	else would you need to add to the register to make	11	what is labeled as CPU and a block is R-E-G-S.
12	it an arbiter?	12	Do you see that?
13	A. Oh, this is a lot of things.	13	A. Yes.
14	Q. Some sort of logic to select, right?	14	Q. That stands for register?
15	A. Yeah. You asked me outside the scope of	15	A. Yes.
16	this case. Are you asking me a general question	16	Q. Is that register that you're showing in
17	about computer architecture? Because I want to know	17	your diagram an arbiter?
18	what the relevance is to what my report is and so	18	A. Not in that case.
19	on. If you could kindly direct me to what I'm	19	Q. Because it doesn't select?
20	prepared to talk to you about here. I mean, I can	20	A. That one is used for holding temporary
21	talk about these things if I have to because that's	21	data for the PC for the CPU. I get confused that
22	my job to answer your questions, but I don't see the	22	PC is program counter.
23	relevance to this.	23	(WHEREUPON, document marked as
24	 Q. Earlier you said a register can read and 	24	Exhibit No. 7.)

1	Page 70		Page 72
	BY MR. TUMINARO:		BY MR. TUMINARO:
2	Q. You have been handed what has been marked	2	Q. Let me clarify. The system disclosed in
3	as Exhibit 7 for your identification. This is an	3	Stuttard does geometry processing?
4	U.S. Patent 7363427. This is the Stuttard patent.	4	A. Let me check.
5	A. Yes.	5	Yes.
6	Q. You have reviewed this patent?	6	Q. Okay. And the system disclosed in
7	A. Yes, sir.	7	Stuttard also does rasterization or pixel-type
8	Q. It's listed on your materials considered?	8	operations?
9	A. Correct.	9	A. Correct.
10	Q. If you look at figure 3 of Stuttard, what	10	O. In fact, Stuttard discloses that the
11	is shown in figure 3 what's labeled as a processing	11	geometry operation and rasterization operations
12	block 106.	12	happen in phases, right?
13	Do you see that?	13	MR. PLUTA: Object to form.
14	A. Yes.	14	BY THE WITNESS:
15	Q. And each of these processing blocks	15	A. Can you show me what you're talking
16	include a plurality or an array of processing	16	about?
17	elements 1061?	17	BY MR. TUMINARO:
18	A. Yes.	18	Q. Sure. If you turn to column 18. I think
19	Q. If you turn with me to column 4, line 33,	19	it is line 11. I'll read it for you.
20	it reads in the example shown in Figure 2, "the	20	A. Do you mind if I read it on my own and
21	processing core 10 is provided with eight processing	21	then you can read it for the record. I would prefer
22	blocks 106."	22	that, if you don't mind.
23	Do you see that?	23	Q. Sure.
24	A. Sorry. Column four, line what?	24	A. Go ahead.
	Page 71		Page 73
1	Q. 33.		
2			~
	A. 33. Core 10? Processing core 10 is		A Thombreau
	marridad with aight measaging bloaks 106	2	A. Thank you.
3	provided with eight processing blocks 106.	3	Q. "The bidding process must maintain
3 4	Processing core 10. Where is processing core 10?	3 4	Q. "The bidding process must maintain primitive order between the geometry and
3 4 5	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to	3 4 5	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most
3 4 5 6	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you.	3 4 5 6	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems."
3 4 5 6 7	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you.	3 4 5 6 7	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that?
3 4 5 6 7 8	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir?	3 4 5 6 7 8	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly.
3 4 5 6 7 8	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes.	3 4 5 6 7 8	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry
3 4 5 6 7 8 9	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left,	3 4 5 6 7 8 9	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right?
3 4 5 6 7 8 9 10	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion	3 4 5 6 7 8 9 10	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these
3 4 5 6 7 8 9 10 11	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks.	3 4 5 6 7 8 9 10 11	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean.
3 4 5 6 7 8 9 10 11 12	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that?	3 4 5 6 7 8 9 10 11 12	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right?
3 4 5 6 7 8 9 10 11 12 13	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes.	3 4 5 6 7 8 9 10 11 12 13	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it.
3 4 5 6 7 8 9 10 11 12 13 14	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown	3 4 5 6 7 8 9 10 11 12 13 14	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what
3 4 5 6 7 8 9 10 11 12 13 14 15	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks?	3 4 5 6 7 8 9 10 11 12 13 14 15	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right?
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay.	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay. Q. Right. So in this example, there are	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase means here. We can look at it if you want.
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay. Q. Right. So in this example, there are eight processing blocks; is that right?	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase means here. We can look at it if you want. Q. Sure. The next sentence in column 8, it
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay. Q. Right. So in this example, there are eight processing blocks; is that right? A. Eight 106s, that's correct.	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase means here. We can look at it if you want. Q. Sure. The next sentence in column 8, it says since both phases, referring back to the
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay. Q. Right. So in this example, there are eight processing blocks; is that right? A. Eight 106s, that's correct. Q. Now, Stuttard says that the well,	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase means here. We can look at it if you want. Q. Sure. The next sentence in column 8, it says since both phases, referring back to the geometry and rasterization phases, are block
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay. Q. Right. So in this example, there are eight processing blocks; is that right? A. Eight 106s, that's correct. Q. Now, Stuttard says that the well, strike that.	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase means here. We can look at it if you want. Q. Sure. The next sentence in column 8, it says since both phases, referring back to the geometry and rasterization phases, are block parallel, there needs to be a mechanism for
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	Processing core 10. Where is processing core 10? Q. If you look at figure 2B. I'll try to help you. A. Thank you. Q. Are you there, sir? A. Yes. Q. You see it sort of toward the top left, there's a label 106, and it says eight times fusion blocks. Do you see that? A. Yes. Q. And that 106 is the same number as shown in figure 3, the processing blocks? A. Okay. Q. Right. So in this example, there are eight processing blocks; is that right? A. Eight 106s, that's correct. Q. Now, Stuttard says that the well,	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	Q. "The bidding process must maintain primitive order between the geometry and rasterization phases due to the requirement of most host systems." Do you see that? A. Yes. You read it correctly. Q. That means Stuttard discloses a geometry phase as distinct from a rasterization phase, right? A. It says phases. I don't know what these phases mean. Q. You reviewed Stuttard, right? A. I didn't memorize it. Q. You provided an opinion about what Stuttard discloses, right? A. I did. But I don't recall what the phase means here. We can look at it if you want. Q. Sure. The next sentence in column 8, it says since both phases, referring back to the geometry and rasterization phases, are block

Page 74 Page 76 1 Do you see that? 1 Q. Let's look further down at line 29. "A 2 A. Yes. 2 record is kept of how many primitives are written to 3 3 Q. When it says that the phases are block each bin so that regions can be sorted into similar parallel, that means all the processing blocks 106 4 size groups for block parallel rasterization." 5 5 do geometry processing during the geometry phase and Do you see that? 6 6 all the processing blocks 106 do pixel processing A. Yes. 7 7 during the rasterization phase, right? Q. Block parallel rasterization means all 8 MR. PLUTA: Object to form. 8 blocks are doing rasterization, right? 9 9 BY THE WITNESS: MR. PLUTA: Object to form. 10 A. I could not tell you that. 10 BY THE WITNESS: 11 BY MR. TUMINARO: 11 A. Yes. I think you're taking this out of 12 Q. Well, let's go down then to line 18. It 12 the original context. Block parallel doesn't mean 13 says: "This is implemented by creating multiple 13 that there's no other activities going on. That 14 14 bins -- strike that. Let me start again. means there's a data parallelism in the computation 15 "This is implemented by creating 15 meaning that you apply a single instruction to 16 multiple bin lists per region, one for every 16 multiple data to some extent. So I think you're 17 processing block 106 that is processing geometry 17 taking a block parallel out of its original context, 18 18 data. This allows the geometry output phase to which means multiple things could happen at the same 19 19 proceed in block parallel mode." 20 Do you see that? 20 So, I mean, I'll be glad to look at 21 A. You're reading it correctly. 21 the phases and so on, but that's not the way I read 22 22 Q. That means the geometry phase is this. To have more information, I mean I have to 23 performed in block parallel mode, right? 23 look at the phases. 24 2.4 MR. PLUTA: Object to form. I don't want to take too much time to Page 75 Page 77 1 BY THE WITNESS: explain what the block parallel is, but it has to do 2 A. You're reading what it says or you're 2 with data parallelism, and I think that means the 3 3 asserting what it says. processing elements are working, all of them, on one 4 BY MR. TUMINARO: 4 block, like you said, but there are multiple 5 5 processes that could do that. Q. So what that means is that block 6 parallel, during the geometry phase, all processing 6 Q. In fact, in Stuttard, there is one core 7 7 blocks 106 are doing geometry operations; right? processor, that is core processor 10, right? Look 8 8 at figure 2B. MR. PLUTA: Object to form. 9 9 BY THE WITNESS: 2B. Okay. Yes. 10 10 A. I could not tell you that. The phases And that core processor has eight blocks, Q. 11 11 right? could happen at the same time or in different levels 12 12 of -- you know, this is very detailed analysis of A. Yes. 13 what Stuttard does. 13 Q. And column 18 says that geometry output 14 14 BY MR. TUMINARO: phase proceeds in block parallel mode, talking about 15 15 Q. But Stuttard says they occur in block all the blocks 106 on core 10? 16 16 A. Yes. But figure 3 shows you -- you see parallel, right? 17 the dot dots on the processor unit? Inside the MR. PLUTA: Object to form. 17 18 18 BY THE WITNESS: belly of 106, there are whole bunch of processors 19 19 A. That has nothing to do with phases. That inside that. You have got eight times whatever that 20 20 is. I have to read more to figure out how the block means it is data parallel. 21 BY MR. TUMINARO: 21 is done. Is it done across this or inside this? I 22 22 Q. It says there is geometry phases. would be glad to look at all of this if you want. 23 23 No. Block parallel means other things. But I disagree with your statement there. 24 Data parallel is a different business. 24 Q. Let's look at what each processing

Page 80 Page 78 element within a block does. If you'll turn with me the ME for processing in to the pixel data." 1 1 2 2 to column 9. Do you see that? 3 3 A. Sure. A. Can you tell me the column again? 4 4 Q. Column 10, line 30. Q. Line 57. 5 5 A. Okay. Thanks. 6 Q. It says, "the data is loaded into the PEs 6 Do you see that? 7 7 of the graphic system so that each PE contains data A. Yes. 8 for one vertex." Do you see that? 8 Q. The last sentence in that paragraph, line 9 9 A. Which PE are we talking about? 39, it says: "Each PE also includes data about a 10 Q. Those are the PEs that are inside a block 10 respective pixel (ie, data is stored on a pixel per 11 106 if you look at the figure 3. 11 PE basis)." 12 A. Okay. 12 Do you see that? 13 13 Q. There's processing elements 1061. A. You're reading it as it says, yes. 14 A. I understand. 14 Q. In line 45 -- well, I'll read line 42. 15 15 "Once each pixel is determined to be outside or Q. You agree that at column 9, paragraph --16 line 57, Stuttard is saying that each of the 16 inside the triangle (primitive) concern, the 17 processing elements stores vertex data, right? 17 processing for the primitive can be carried out only 18 18 MR. PLUTA: Object to form. on those pixels occurring inside the perimeter." 19 19 BY THE WITNESS: Do you see that? 20 A. Reading that, it says each -- the data is 20 A. You're reading it correctly, yes. 21 loaded into the PEs of the graphic system so that 21 Q. And then it goes on to say that "the 22 each PE contains the data for one vertex. That I 22 remainder of the PEs in the processing block do not 23 23 take any further part in the processing until that understand. 2.4 24 primitive is processed." Page 79 Page 81 1 BY MR. TUMINARO: 1 Do you see that? 2 Q. So each PE has vertex data? 2 A. Yes. 3 3 Q. That means during the pixel operation, A. One of the things it does, it has the 4 vertex data among other things. 4 all the PEs are doing pixel operations or nothing, 5 Q. Each PE in block 106 has vertex data 5 right? 6 according to what Stuttard is saying? 6 MR. PLUTA: Object to form. 7 A. For this example, yes, but it could have 7 BY THE WITNESS: 8 8 A. With the caveat that there are eight other things. 9 9 Q. You. But Stuttard doesn't say that, slices of those, and other slices could be doing 10 right? Stuttard said each PE has vertex data. 10 other things. You just looked at one slice. 11 11 BY MR. TUMINARO: A. In that sentence, it says that. But I 12 12 have to look at the rest to see if it does other Q. Okay. 13 things. 13 A. I agree with what you said about one 14 14 Q. It continues on to say "that each PE then slice. Across -- let me finish, please. 15 15 represents a vertex of a primitive that could be at MR. PLUTA: You guys are talking over 16 an end of a line or part of a two-dimensional shape 16 each other. 17 such as a triangle." 17 THE WITNESS: I'm trying to finish my 18 18 A. For that paragraph, that is correct. explanation if you don't mind. There are eight 19 19 But, again, PEs could be doing other things. slices, and they could be doing other things 20 Q. Then let's look at what Stuttard says 20 including vertex. 21 about pixel processing. Let's look at what it says 21 BY MR. TUMINARO: 22 22 about pixel processing. Q. But each slice, when it's doing pixel 23 23 operations -- the slice you're talking about at At column 10, line 30, "each PE then 24 transfers in its data concerning its primitive to 24 block 106, right? When it's doing pixel operations,

Page 82 Page 84 1 every processing element within block 106 does 1 are doing vertex operations when block 106 is doing 2 2 either a pixel operation or nothing, right? a vertex operation? 3 3 A. Can I -- not having read the whole thing MR. PLUTA: Object to form. 4 again here, by looking at what you showed me that 4 BY THE WITNESS: 5 5 you are correct, but I want to qualify that that I A. Okay. It doesn't say it has the data for 6 need to check the whole document to make sure that 6 one vertex only. It could have data for pixels 7 it doesn't say another embodiment we can do these. 7 already in the register. So I cannot say from this 8 8 I want to be clear on that. statement that it does not have other information. 9 Q. But here, that's what this means? 9 If it said only vertex, you're right. 10 10 A. In this example as you provided -- if BY MR. TUMINARO: 11 it's one embodiment, you're right. But if the other 11 Q. It does say each PE, right? 12 embodiments are there, I'm not agreeing to what you 12 A. No. 13 13 say. Q. I'll read it again. 14 Q. Okay. So that's for pixels. 14 A. I didn't -- if I could qualify what I 15 15 And then for vertices, within a said. 16 16 processing block, each processing element within Q. Let me read it again for the record. 17 that processing block does vertex operations, right? 17 A. 18 18 A. As I tried to explain, it could be that "The data is loaded into the PEs for the 19 19 each slice is doing different things at the same graphic system so that each PE contains data for one 20 time -- at the same time. 20 vertex." 21 Q. I appreciate that. But what I'm trying 21 Did I read that correctly? 22 to get out now is what a particular slice or 22 A. You did. It's not convincing to me that 23 processing block is doing. And if a processing 23 it's the only thing that it has. 24 block is doing pixels, all the processing elements 24 Q. You don't read each PE to mean that every Page 83 Page 85 1 within that processing block do pixels or nothing, 1 single PE in the block has vertex data? 2 2 we established that, right? A. No, Counsel. It should have said that 3 3 A. For this embodiment. only vertex data, then I agree with you. It doesn't 4 Q. Now, I want to talk for vertices. If a 4 say only. Does it say? 5 5 Q. It says each PE -processing block is doing vertices, all the 6 processing elements within that block do vertices, 6 A. Right. 7 7 right? Q. -- has vertex data. So you're saying a 8 8 PE could have other data? A. Can you show me that section because you 9 9 showed me for the pixel, not vertices. A. Absolutely. From previous computations. 10 10 Q. Let's look for the vertices. It actually does something like that because you 11 11 need to keep track of the state of what you did. Column 9, line 57, "the data is 12 12 The registers -- you showed me the register file. loaded into the PEs of the graphic system so that 13 each PE contains data for one vertex." 13 They have registers. They keep track of the state. 14 14 That means every PE has vertex data, MR. TUMINARO: Why don't we take a break. 15 15 THE VIDEOGRAPHER: We're going off the right? 16 MR. PLUTA: Object to form. 16 record. This is the end of Media 2. The time is 17 BY MR. TUMINARO: 17 11:15 a.m. 18 1.8 Q. Did you nod your head? Is that a yes? (Whereupon, a break in the 19 19 A. Sorry. Apologies. I'm reading and proceedings was taken.) THE VIDEOGRAPHER: We're back on record. 20 20 listening to you at the same time. 21 Okay. Yes. You're reading what it 21 This is the beginning of Media 2. The time is 22 22 11:36. says, yes. 23 23 Q. Well, just to be clear, that means that BY MR. TUMINARO: 24 all the processing elements within that block 106 24 Q. Welcome back, sir.

Page 88 Page 86 1 1 But the understanding of unified A. Thank you. 2 2 You have heard of the phrase unified shader in general, it's a programmable system that 3 3 shader? can handle multiple functions as part of shading. 4 4 That's the understanding I have. A. Yes. 5 5 Q. And unified shader refers to an BY MR. TUMINARO: 6 architecture where the same hardware does both 6 Q. Does claim 1 handle vertex operations and 7 7 vertex operations and pixel operations, right? pixel operations? 8 MR. PLUTA: Object to form. 8 MR. PLUTA: Object to form. 9 9 BY THE WITNESS: BY THE WITNESS: 10 10 A. Claim 1 says that an arbiter -- okay. A. That's the general idea. 11 BY MR. TUMINARO: 11 Does it handle vertex operations? It refers to 12 Q. The claims of the '053 patent are 12 vertex data. I agree with that. It says that it 13 directed to a unified shader, right? 13 will select between vertex and pixel data. That, I 14 A. Did they use the word unified shader 14 see and I agree with the statement here. 15 15 anywhere? Can you point me to it? BY MR. TUMINARO: 16 Q. I guess, if you look at claim 1, it is 16 Q. If you would turn with me with 17 talking about a single set of hardware that does 17 Paragraph 19 of your declaration. 18 18 both vertex operations and pixel operations, right? A. Paragraph 19, okay. 19 19 A. Where do you get that information from Q. Page 5 if that helps. It reads, "I also 20 claim 1? 20 understand that the relevance" -- excuse me. I'll 21 Q. Well, the memory stores both vertex 21 start again. 22 commands and pixel demands, right? 22 "I also understand that the relevant 23 23 A. That's the memory. inquiry into obviousness requires consideration of 2.4 24 And the arbiter selects either vertex four factors." Page 87 Page 89 1 commands or pixel commands, right? 1 Do you see that? 2 A. Correct. 2 A. Yes. 3 3 Q. It's a single set of hardware that's O. And one of the factors that's listed as 4 doing --4 (d) is objective factors, right? 5 A. Overall, it's hardware with multiple 5 A. Correct. 6 6 Q. And one of the objective factors is functional units. 7 7 Q. It's doing vertex operations and pixel copying of the invention by others in the field? 8 operations? 8 A. Commercial success, long-felt needs, 9 9 A. It's really a broad definition, unified copying of the invention by others in the field. 10 shader. You could have specialized hardware, but I 10 That's what it says, yes. 11 11 mean, I haven't really opined on that for this case. Q. You would agree with me that copying of 12 12 the invention by others is objective evidence of Q. I guess I'm asking you right now as an 13 expert in this field, would you consider claim 1 to 13 nonobviousness? 14 14 be directed to a unified shader? MR. PLUTA: Object to form. 15 15 MR. PLUTA: Objection, relevance. BY THE WITNESS: 16 Objection, form. 16 A. Okay. You are asking me -- I know it's 17 BY THE WITNESS: 17 in my report, but this is a legal matter. My level 18 A. I haven't formed an idea on this because 18 of understanding, it is exactly what it says here. 19 19 it's really not a matter of just being a I cannot dwell on this more than what it is. You 20 programmable shader. One has to see more details 20 know, I'm not here for legal issues; but you're 21 about it. I couldn't tell you for certain that this 21 right, it is in my report and that's what it is. I 22 22 is a unified shader because there are no more cannot add any more than what you see there 23 23 unfortunately. Sorry. details about what these functional units do. I 24 cannot say from what I have here. 24

Page 90 Page 92 1 BY MR. TUMINARO: 1 BY MR. TUMINARO: 2 2 Q. And computer graphics chips have to be Q. Okay. You're aware, as an expert, that 3 compatible with DX10 if they want to work on that 3 nearly the entire computing industry now uses 4 4 unified shader? platform, right? 5 5 MR. PLUTA: Objection, form. Objection, MR. PLUTA: Object to form. Objection, 6 relevance. 6 relevance. 7 7 BY THE WITNESS: BY THE WITNESS: 8 A. Which computer graphic chips? There are 8 A. Yeah. I could not tell you that. I have 9 9 not opined on that for my report. It requires a whole bunch of computer graphic chips. 10 10 looking at the entire industry. BY MR. TUMINARO: 11 BY MR. TUMINARO: 11 Q. Any graphic chip that's want to be 12 Q. You have heard of the DX10? 12 compatible with DX10 would have to implement what 13 MR. PLUTA: Object to form. Sorry. 13 DX10 talks about, right? 14 Withdraw the objection to form. But objection to 14 MR. PLUTA: Same objection. I'm sorry, 15 15 Counsel. That prior objection also included a relevance. 16 BY THE WITNESS: 16 relevance objection. 17 A. I have heard of it. 17 BY THE WITNESS: 18 18 A. Without having studied it, if I say BY MR. TUMINARO: 19 19 Q. It's an application programming interface something is able to do a particular application or 20 developed by Microsoft? 20 API, then I have to be able to do it. I can answer 21 MR. PLUTA: Objection, relevance. 21 that in a general term. I would not advertise that 22 BY THE WITNESS: 22 I could do graphics for DX10 API and not be able to 23 23 do it. A. You are asking me something outside the 24 24 scope of my report. It is an API, but that's all I Page 91 Page 93 1 can tell you about that. I need to evaluate it BY MR. TUMINARO: 2 more. 2 Q. Let me ask this, I guess. 3 BY MR. TUMINARO: 3 A. Sure. 4 Q. I'll just say for the record your report 4 DX10 includes a unified shader mode, 5 talks about objective indicia and copying is 5 right? 6 relevant inquiry into obviousness. You opined on 6 MR. PLUTA: Objection, form. Objection 7 7 obviousness, right? relevance. 8 MR. PLUTA: Objection, form. 8 BY THE WITNESS: 9 BY THE WITNESS: 9 A. I have not looked at that. 10 10 A. I did opine on obviousness to the level MR. TUMINARO: I'll have this marked. 11 11 (WHEREUPON, document marked as of an expert using it to evaluate the prior art, but 12 secondary indices or whatever you call it, that's 12 Exhibit No. 8.) 13 not my specialty. I will not be able to give you 13 BY MR. TUMINARO: 14 any meaningful example, although I understand some 14 Q. You have been handed what has been marked 15 15 as Exhibit 8. This is the DX10 architecture. of it. 16 BY MR. TUMINARO: 16 Do you see that? 17 Q. As an expert -- you claim to be an expert 17 18 18 MR. PLUTA: I'm going to object to the in computer graphics? 19 19 introduction of this exhibit as irrelevant. Also A. Yes. 20 Q. You have heard of DX10? 20 hearsay. 21 MR. PLUTA: Objection, relevance. 21 BY MR. TUMINARO: 22 22 BY THE WITNESS: Q. If you will turn with me to Page 14. A. Yes. 23 23 Page 14 at the top says DX10 shader model 4.0. 24 24 Do you see that?

Page 94 Page 96 1 A. Yes. 1 then later on. 2 2 After it reads "shader model 4.0 (SM 4.0) Q. You don't know one way or the other 3 is the new instruction set architecture (ISA) for DX 3 whether DX10 or its later version require a unified 10 that looks at the graphics in a unified way." 4 shader? 5 5 Do you see that? MR. PLUTA: Objection, form. Objection 6 A. Yes. 6 relevance. 7 Q. If you look at the second bullet, it 7 BY THE WITNESS: 8 8 says, "flexible load balancing." A. If I read the documents, I can tell you 9 Do you see that? 9 10 10 A. Yes. BY MR. TUMINARO: 11 11 Q. And the third sentence says that "the Q. Why don't you read it. 12 unified shader is made up of shader blocks that 12 A. If I need additional documents, can I get 13 could handle all vertex, pixel, and geometry 13 them if it's not sufficient? 14 instructions." 14 Q. Why don't read the document and we'll go 15 15 Do you see that? from there. 16 16 A. Yes. You're reading the sentence. A. Okay. 17 Q. Okay. So if a GPU wants to be compatible 17 MR. PLUTA: While he's doing that, I'm 18 18 with DX10 shader model 4.0, the GPU has to offer a just going to also lodge an objection to the 19 19 unified shader, right? authenticity of this document. 20 MR. PLUTA: Objection, relevance. 20 In addition to the authenticity, I'm 21 Objection, hearsay. 21 going to object also to the foundation with the 22 BY THE WITNESS: 22 introduction of this exhibit. 23 23 A. I could not tell you that from that THE WITNESS: The question was on 14. 24 2.4 sentence. You picked one sentence out of this Okay, your question is? Page 95 Page 97 1 document. I'm a very precise person. If I have to BY MR. TUMINARO: 2 make a comment on that, I cannot do that with one 2 Q. After reviewing this --3 sentence. You lifted one sentence out of this 3 A. Up to here. I haven't finished. Do you 4 document that I have never seen before. 4 want me to finish? 5 5 Q. Page 14? BY MR. TUMINARO: 6 Q. Have you used DX10 shader model 4.0? 6 A. Yeah. 7 7 MR. PLUTA: Objection, form and Q. You can look at Page 14. 8 8 A. No. I haven't finished the whole relevance. 9 9 BY THE WITNESS: document. I stopped at 14. Do you want me to read 10 10 more? A. No, I have not. 11 11 BY MR. TUMINARO: Q. I guess -- let's see if you can answer my 12 12 question. Q. Have you used any of the later versions 13 of DX, DX11, DX12 offered by Microsoft? 13 After reviewing this, does DX10 14 14 MR. PLUTA: Objection, relevance. shader model 4.0 require a unified shader? 15 15 MR. PLUTA: Objection, form. Objection Objection, form. 16 BY THE WITNESS: 16 relevance. 17 A. I don't know how it's related to my 17 BY THE WITNESS: 18 report. 18 A. Although it is not related to my report 19 19 BY MR. TUMINARO: and I had a quick review of this document, it 20 Q. You are an expert in graphics, right? 20 provides a programmable shader. 21 A. Yes. 21 BY MR. TUMINARO: 22 22 Q. Supposedly. Q. A unified shader? 23 23 I do not appreciate the comment about MR. PLUTA: Objection, form. Objection 24 supposedly. But I have not used this one, no, and 24 relevance.

Page 98 Page 100 1 BY THE WITNESS: BY THE WITNESS: 1 2 2 A. A computing resource that can be A. I could not tell you. That's a legal 3 programmed to do different shading. 3 issue. I cannot opine on that. I'm sorry. I mean 4 BY MR. TUMINARO: 4 I know I have it in my report. It is just to tell 5 5 Q. So before you submitted your declaration, you that I know about this information, but I cannot 6 you didn't know one way or the other whether DX10 6 go beyond what this sentence says. Analyze it or 7 required a unified shader? 7 discuss it in any way. 8 MR. PLUTA: Objection, form. Objection 8 BY MR. TUMINARO: 9 relevance. 9 Q. Okay. In your report, you didn't analyze 10 10 BY THE WITNESS: any secondary factors of nonobviousness, right? 11 A. I'm very familiar with unified shaders. 11 MR. PLUTA: Objection, form. 12 BY MR. TUMINARO: 12 BY THE WITNESS: 13 Q. But my question is did you know that DX10 13 A. I don't recall I did that. If I'm 14 required a unified shader? 14 missing something, please let me know. I don't 15 MR. PLUTA: Same objections. 15 recall doing that. 16 BY THE WITNESS: 16 BY MR. TUMINARO: 17 A. No, I did not. 17 Q. In this case, counsel provided you with 18 18 BY MR. TUMINARO: all the prior art that you considered, right? 19 Q. You have heard of open GL? 19 MR. PLUTA: Objection, form. 20 A. Yes. 20 BY THE WITNESS: 21 Q. Before you submitted your declaration, 21 A. I identified prior art myself. 22 did you know one way or the other whether the 22 BY MR. TUMINARO: 23 current version of open GL required the unified 23 Q. You identified the prior art that you 24 24 shader? applied? Page 99 Page 101 1 MR. PLUTA: Objection, form. Objection 1 A. Right. 2 2 Q. Did Counsel provide you with any evidence relevance. 3 3 of objective indicia of nonobviousness? BY THE WITNESS: 4 A. I actually knew about open GL and unified 4 MR. PLUTA: Objection, form. Objection, 5 shaders. And now I recall that DX. I also knew 5 relevance. 6 about unified shaders for another. 6 BY THE WITNESS: 7 BY MR. TUMINARO: 7 A. Again, I cannot comment on that because 8 Q. So they both required unified shader? 8 I'm not familiar with the exact term in legal. 9 9 At that time I knew, yes. BY MR. TUMINARO: 10 MR. PLUTA: Objection, form. Objection 10 Q. Did you go out and try to see if anyone 11 11 was copying the claimed invention? relevance. 12 12 BY MR. TUMINARO: MR. PLUTA: Objection, form. Objection, 13 Q. In Paragraph 19 you mentioned -- of your 13 relevance. 14 14 declaration. BY THE WITNESS: 15 15 A. Going out means what? Going visiting A. Okay. 16 Q. You mentioned objective factors are one 16 companies and looking at chips? 17 of the things that must be required for an 17 BY MR. TUMINARO: 18 18 obviousness analysis. Q. Did you do any researching, any analysis, 19 19 any inquiry to see if there was copying of the A. Okay. I say that. And the question is? 20 Q. Well, that's true, right? Objective 20 claimed invention? 21 indicia is one of the things that must be analyzed 21 MR. PLUTA: Same objections. 22 22 in an obviousness consideration? BY THE WITNESS: 23 MR. PLUTA: Objection, form. 23 Copying of the claimed inventions. 24 24 Outside what I have in this report, I have nothing

Page 102 Page 104 1 MR. PLUTA: Objection, form, objection 1 else to add. BY MR. TUMINARO: 2 2 relevance. 3 3 Q. So the answer is you didn't do that? BY THE WITNESS: 4 A. To see if anybody has copied the claims 4 A. I can't comment on that, had to. There 5 5 for the product? might be other options. 6 6 MR. PLUTA: Objection, form. BY MR. TUMINARO: 7 7 BY THE WITNESS: Q. If they want to be compatible with those 8 8 A. No. What is in this report is what I APIs, they would have to include it, right? 9 9 have done. MR. PLUTA: Same objections. 10 BY MR. TUMINARO: 10 BY THE WITNESS: 11 Q. You didn't do any analysis about 11 A. It requires a thorough assessment. I 12 long-felt need for the claimed invention? 12 cannot comment on that. 13 A. It's not in this report, correct? 13 BY MR. TUMINARO: 14 Q. That's correct. 14 Q. What assessment would you have to do to 15 15 A. Whatever is in this report I have done. determine what --16 Q. You didn't do any analysis of failed 16 MR. PLUTA: Objection, form. Objection 17 attempts by others to create the claimed invention? 17 relevance. 18 18 A. Again, I don't recall; but if it's here, BY THE WITNESS: 19 19 it's in the report. A. Not during a deposition. I have to spend 20 Q. The fact is you didn't consider any 20 time to look into it, the details and so on. I will 21 objective indicia of nonobviousness in forming your 21 not be able to opine on that right here based on 22 opinion about obviousness; is that right? 22 what you showed me. 23 23 MR. PLUTA: Objection, form. BY MR. TUMINARO: 24 24 Q. And you didn't consider that -- any of Page 103 Page 105 that in forming your opinions on obviousness? 1 BY THE WITNESS: 2 A. That's a, again, legal matter. I will 2 MR. PLUTA: Objection, form. 3 not want to opine on something I'm not very familiar BY THE WITNESS: 3 4 with, but I mean, it says what it says. 4 A. Whatever I opined on is right here. 5 5 BY MR. TUMINARO: MR. TUMINARO: Let's break for lunch. 6 Q. You considered only prior art in forming 6 MR. PLUTA: Sure. 7 7 your opinions on obviousness? THE VIDEOGRAPHER: We're going off 8 MR. PLUTA: Objection, form. 8 record. The time is 12:06. 9 9 BY THE WITNESS: (Whereupon, a short break in the 10 10 A. I considered what's in the exhibits, proceedings was taken.) 11 11 THE VIDEOGRAPHER: We're back on record. whatever you see here. Prosecution history, patents 12 The time is 12:50. and so on. These are the items I considered. I 12 13 think we went through that with you. If I had done 13 BY MR. TUMINARO: 14 14 something, it would be here. Q. Welcome back, sir. You testified earlier 15 15 I believe that you designed GPU? BY MR. TUMINARO: 16 Q. So now you remember now that DX10 and 16 A. I designed a processor that had GPU 17 open GL require a unified shader? 17 capabilities. 18 18 Q. What's the first step that you took in MR. PLUTA: Objection, relevance. 19 BY THE WITNESS: 19 designing that processor that had GPU capabilities? 20 20 MR. PLUTA: Objection, form. Objection A. Yes, I looked at that before, yeah. 21 BY MR. TUMINARO: 2.1 relevance. 22 22 Q. So if a graphics processing architecture BY THE WITNESS: 23 23 A. You are asking me a general question, not wants to be compatible with either of those APIs, 24 the GPU would have to have a unified shader, right? 24 related to this case, right? Because I haven't

Page 106 Page 108 1 opined about that particular question in this 1 of an C code or something? 2 2 MR. PLUTA: Objection, form. Objection, document. 3 BY MR. TUMINARO: 3 relevance. 4 4 BY THE WITNESS: Q. What was the process -- what was the 5 5 first step that you took --A. Yeah. Right. 6 MR. PLUTA: Same objections. Sorry. 6 BY MR. TUMINARO: 7 BY MR. TUMINARO: 7 Q. After you did the simulations of the C 8 8 Q. -- in developing the chip that had a GPU code, that's when you went to the VHDL? 9 capability? 9 MR. PLUTA: Same objections. 10 10 BY THE WITNESS: A. Coming up with a new idea. Epiphany. 11 You take the idea to the process of simulating it, 11 A. Yes. That's what we do. That's what we 12 you simulate using some sort of a cycle accurate 12 do. That's what we did then, and we continue to do 13 simulator. 13 the same thing. 14 Q. You said? 14 BY MR. TUMINARO: 15 A. Cycle accurate simulator. 15 Q. Is that what is done in the industry? 16 Q. Cycle accurate, is that what you said? 16 A. It is. 17 A. Yes. And then once you're happy with the 17 MR. PLUTA: Objection, form. Objection, 18 18 results, you move onto the level of, like we talked relevance. 19 19 about, writing the VHDL code. BY MR. TUMINARO: 20 Q. You simulate it. What are you simulating 20 Q. Is that what you teach your students? 21 then at that point? 21 MR. PLUTA: Objection, relevance. 22 MR. PLUTA: Objection, form. Objection 22 BY THE WITNESS: 23 23 A. Yeah. That's what we do. We not only we relevance. 2.4 24 teach our students, we also -- we continue our Page 109 Page 107 1 BY THE WITNESS: 1 research in that direction. It would be unwise to 2 A. The ideas. The architectural ideas. 2 go and design the hardware without having it 3 BY MR. TUMINARO: 3 modeled. And that's true for every field I would 4 Q. I guess what I'm confused about, what I 4 say, not just us. And I think -- I would hope not 5 don't understand is what form are the architectural 5 thinking that it is my personal style. It's the 6 ideas in? Are they on paper? Are they in your 6 style of all the designers do -- go for it. I mean, 7 head? 7 they basically model simulate. Then they go to the 8 MR. PLUTA: Objection, form. Objection, 8 hardware. 9 9 relevance. BY MR. TUMINARO: 10 10 BY THE WITNESS: Q. What does the model tell you? 11 A. Well, I mean, they go from your head to 11 MR. PLUTA: Objection, form. Objection, 12 the piece of paper, and then finally you write the 12 relevance. 13 software to represent the architecture. 13 BY THE WITNESS: 14 14 Let's use the morphosys as an example A. If you recall, I said cycle accurate. So 15 15 since you had done your due diligence and looked at it gives you the number of cycles it will take to do 16 that paper. We had to simulate it before we go into 16 a computation. 17 fabricating it. We had funding from the government 17 So, the morphosys project, we had a 18 18 agencies. It's unusual for a university to actually cycle accurate or close to cycle accurate model of 19 19 fabricate a chip as you probably were alluding to. the system. We massaged certain things. 20 But we had funding. We went simulations -- we did 20 BY MR. TUMINARO: 21 simulations and then ultimately we went to the VHDL 21 Q. What does that mean? 22 22 model and did the layout. MR. PLUTA: Objection, form. 23 BY MR. TUMINARO: 23 BY THE WITNESS: 24 Q. Oh, okay. You simulated, what, some kind 24 A. Meaning that we had -- I don't know your

Page 110 Page 112 1 background but engineering is all about tradeoffs 1 combination thereof such that it receives the 2 2 command thread and thereupon provides the command and, you know, we're coming up with new things 3 3 instead of finding out what is already there. Like, thread to a command processing engine." 4 what physicists and chemists will do, they find 4 So that's my definition. 5 5 things that are already there. We're trying to come BY MR. TUMINARO: 6 6 up with new things. There's a lot of trial and Q. What part of that definition describes 7 7 error. So that's why when you model and simulate that the arbiter selects from available inputs? 8 8 A. It basically describes how the arbiter is based on the application, that application presents 9 9 itself, it says this particular piece of implemented, and arbiter as we described is the one 10 10 architecture is not the right choice for me. We go that selects between inputs. 11 and massage it and go change it. I don't know when 11 Q. Just to be clear, your interpretation of 12 I start whether I have the optimal solution or not. 12 arbiter is in Paragraph 52 of your declaration, 13 13 It is a very trial and error basis. 14 14 Q. By massage, you're saying it helps you A. Yes. That's the implementation of it. 15 determine what the appropriate architecture should 15 This is the definition of it, right. 51 is the --16 be? 16 how you implement it and 52 explains what it does. 17 A. Adjusted. Yeah. Think of it as dials. 17 Q. So where in your definition in 18 18 You used the word cache as an example. So let's use Paragraph 52 does it explain that the arbiter is 19 19 that as an example. You find out the size of the selecting from available inputs? 20 cache is not appropriate, then you change the cache. 20 A. It says it receives and provides a 21 Q. And you could tell that kind of stuff 21 thread. 22 22 from your model of the chip? Q. You're reading receives and provides to 23 MR. PLUTA: Objection, form. Objection, 23 mean selects from available inputs? 24 2.4 relevance. A. Yes. Page 111 Page 113 1 BY THE WITNESS: 1 Q. If you turn with me to the Lindholm 2 A. Yeah. I mean, that's basically what it 2 reference. 3 3 is. A. Okay. 4 BY MR. TUMINARO: 4 At figure 4 in Lindholm, there's the 5 5 Q. Switching topics. I think earlier you thread control buffer 420? 6 6 mentioned that an arbiter selects from available A. Yes. 7 7 options, right? Q. I just want to make sure I understand 8 8 your position. Is it your position that the thread MR. PLUTA: Objection, form. 9 9 control buffer 420 in Lindholm is the claimed memory BY THE WITNESS: 10 10 A. Arbiter is -- the job of arbiter is to device in claim 1 of the '053 patent? 11 11 MR. PLUTA: Object to form. selecting available inputs. 12 12 BY THE WITNESS: BY MR. TUMINARO: 13 O. So, is an accurate definition of a 13 A. In combination with instruction cache. 14 14 MR. TUMINARO: Let's have this marked, arbiter any computer hardware, software, or 15 15 combination thereof that receives and provides a please. 16 command thread? 16 (WHEREUPON, document marked as 17 MR. PLUTA: Objection, form. 17 Exhibit No. 9.) 18 18 BY MR. TUMINARO: BY THE WITNESS: 19 19 Q. Okay. You have been handed what's been A. I think that's the claim construction 20 that I offered, I think. Let me check that, please, 20 marked as Exhibit 9. This is the petition for inter 21 just to make sure that you got it right. 21 partes review of U.S. Patent 00742053. You have 22 22 Reading for the record Paragraph 51, seen this document before? 23 23 A. I have seen it. It's not on my list of "the '053 patent discloses that an arbiter may be an 24 implementation of hardware, software, or a materials considered, but I've seen it, yes.

Page 114 Page 116 1 Q. If you will turn with me to paragraph --1 report. I agree with what's in there. 2 I mean Page 13? 2 Q. You didn't consider any other time frame 3 3 A. Page 13. All right. besides mid-to-late 2003? 4 Q. And the first full paragraph, there's in 4 MR. PLUTA: Object to form. 5 5 the first sentence, after the comma, it reads: BY THE WITNESS: 6 "Lindholm's thread control buffer." Do you see 6 A. What do you mean by that? 7 7 that? BY MR. TUMINARO: 8 8 Q. Well, you considered the relevant time A. Yeah. 9 9 Q. I'll read it for the record. "Lindholm's period for determining what a person of ordinary 10 skill in the art to be -- to have known would be thread control buffer 420 corresponds to the claimed 10 11 memory device." 11 mid-to-late 2003? 12 Do you see that? 12 A. That's what it says. 13 13 Q. That's the time frame you're considering. Yes. 14 Q. You disagree with the petitioner that the 14 My question is: You didn't consider any other time 15 thread control buffer is the memory device? 15 period, right? 16 MR. PLUTA: Object to form. 16 MR. PLUTA: Object to form. 17 BY THE WITNESS: 17 BY THE WITNESS: 18 18 A. It's part of the memory device. It A. Any other time meaning before that time. 19 19 completes the memory device. It is an important It could not be after that time, right? 20 part of the memory device. 20 BY MR. TUMINARO: 21 BY MR. TUMINARO: 21 Q. It could be before. It could be after. 22 Q. So you have a different position than the 22 You didn't consider any other times when determining 23 23 what a person of ordinary skill in the art would petitioner? 2.4 24 have known? MR. PLUTA: Object to form. Page 115 Page 117 1 BY THE WITNESS: 1 MR. PLUTA: Same objection. 2 A. It is consistent with the petitioner in 2 BY THE WITNESS: 3 3 A. Well, if this was filed September, 2003, my opinion. 4 BY MR. TUMINARO: 4 it would be very illogical for me to assume anybody 5 Q. But you will agree that on Page 13 of the 5 after the filing date or after the application 6 petition, it doesn't say anything about the memory 6 filing date, right? So your question is probably 7 device -- the claimed memory device also including 7 before that. It cannot be after that. 8 8 BY MR. TUMINARO: the instruction cache, right? 9 9 MR. PLUTA: Object to form. Q. So you didn't consider after because you 10 10 BY THE WITNESS: think it is illogical? 11 A. It also does not exclude it and it 11 A. I mean, I'm not a lawyer, but I'm 12 12 doesn't say it's the only thing so.... assuming that would not work very well. 13 BY MR. TUMINARO: 13 Q. How about before? Did you consider 14 14 Q. Let's go back to your declaration. And before mid-to-late 2003? 15 15 I'd like to go to Paragraph 22. It's on Page 7. A. Well, the person of ordinary skill in the 16 A. 22. Yes. 16 art should have all those prior art references 17 Q. And the second sentence of that paragraph 17 available to them, and if they were available, it 18 18 reads: "As a result, I will assume the relevant would be sufficient, whatever that time frame is. 19 19 time period for determining what one of ordinary They're taking the knowledge that they have, and 20 skill in the art knew is mid-to-late 2003." 20 they're combining those references. So that's all I 21 Do you see that? 21 can tell you about that. It should be inclusive of 22 22 A. Yes. all those references. 23 23 That's an accurate statement? Q. If you could go with me to Paragraph 116 24 That's what it says. I mean, this is my 24 of your declaration. It's on Page 40. Further, I

Page 118 Page 120 1 guess at the top of Page 40, you have a figure 4 1 primitive can be carried out only those pixels 2 2 from the Stuttard reference, right? occurred inside the primitive. The remainder of the 3 3 A. That is correct. PEs in the processing block do not take any further 4 Q. And if you look at Paragraph 116, the 4 part in the processing until that primitive is 5 5 last sentence says "because cache unit 1024 includes processed." 6 both vertex and pixel command threads, it is the 6 A. I remember that discussion. 7 7 Q. We agreed that that embodiment discloses claimed memory device." 8 8 Do you see that? that in a given processing block all the PEs do 9 9 either pixel operations or nothing, right? 10 10 Q. What's your basis for saying that the MR. PLUTA: Object to form. 11 cache unit 1024 in Stuttard includes both vertex and 11 BY THE WITNESS: 12 pixel command threads? 12 A. I don't believe I agreed with that. 13 13 A. Because it feeds the instructions for the BY MR. TUMINARO: 14 graphics processing all the instructions. 14 Q. You didn't agree to that earlier? 15 Q. What feeds that? 15 A. You can read what's on the record. I 16 16 agreed with that's what you read which was the PEs A. The cache. 17 Q. So how do you know it has both pixel 17 were doing the same thing, and I also interjected 18 18 command threads and vertex command threads? that they could have state from other computations 19 19 that are going on. And I also interjected that A. It's identified as one cache and that's 20 what it does. If there were more than one, it would 20 those slices could be doing different things. 21 have shown it, and also I can check to see what I 21 Q. I guess my question is in this embodiment 22 have in my claim chart. Let me see if I can find 22 that is disclosed at column 10, line 45, the 23 23 you the text. sentence reads that "the remainder of the PEs in the 24 24 I mean, there's just the first part. processing block do not take any further part in the Page 119 Page 121 1 It says the thread manager 102 is shown in more processing until the primitive is processed." So those PEs in the processing block 2 detail in figure 4, and comprises a cache memory 2 3 3 unit 1024 for storing instructions fetch for each do nothing, right? 4 thread. And Stuttard does vertex and pixel so it is 4 A. That's what he's saying for that block. 5 5 Q. And the other PEs in that block in the 6 sentence before that, he says those other PEs are Later on, if you look at Page 43, the 6 top of it, it says, for example, thread zero may be 7 7 doing pixel operations, right? 8 assigned for general system control, thread one A. It says they're doing pixel, yeah. Yes. 9 9 assigned to execute 2D, two-dimensional activities, Q. In column 10 in that paragraph that we're 10 10 and threads 2 through 7 assigned to executing 3D talking about, it starts at line 43 and goes to line 11 11 activities, such as calculating vertices, 47, the first sentence says that the PEs in the 12 12 primitives, or rastering. processing block are doing pixel operations and the 13 The citation is column 5, lines 56 13 second sentence in that paragraph says that pixel --14 14 through 64. the PEs are not doing anything, right? 15 15 MR. PLUTA: Object to form. Q. If you'd turn with me in Stuttard, 16 Exhibit 7. 16 BY THE WITNESS: 17 17 A. The remainder of the PEs. Not the PEs A. Okay. 18 18 Q. We talked about column 10, lines 42 are not doing anything. It just -- that means if 19 19 there are eight of them, maybe four are doing PE and through 47. Do you recall that? 20 20 A. Where -- when did I talk about that? four are not doing anything, although they may have 21 Q. Earlier today, I read this into the 21 residual computation from previous operations 22 22 record. Starting at line 43, "once each pixel is because they may hand it over to the next step. 23 23 determined to be outside or inside the triangle BY MR. TUMINARO:

24

24

(primitive) concerned, the processing for the

31 (Pages 118 to 121)

Q. Okay. Just to be clear, I just want to

Page 122 Page 124 1 make it clear. This paragraph says that within a 1 my pen back. 2 2 processing block, the PEs will either do -- if that A. I will give you your pen back. 3 processing block is doing pixels, the PEs in that 3 MR. PLUTA: Is there a question pending processing block will either do pixel operations or 4 just to refresh? 5 5 nothing. Is that what this paragraph says? MR. TUMINARO: I'll ask the question 6 Counsel, let's identify the processing 6 again. Thank you. 7 blocks so there's no misunderstanding between you 7 BY MR. TUMINARO: 8 and me and the records. So I want it to be clear 8 Q. Okay. Sir, you mentioned that the other 9 what I'm saying. So let's understand for the record 9 blocks 106 could potentially be doing vertex 10 10 that the eight fusion blocks, I'm saying that those operations when a single block is doing a pixel 11 could be doing different things. What you're saying 11 operation, is that right? 12 what's happening inside the fusion block --12 A. Yes, I am reconfirmed my position on that 13 13 by reading the text. O. That's right. 14 A. -- which is, for the record, it's figure 14 Q. Tell me where. 15 15 A. I'll tell you the text and then we can 3, identified as 106 processors. Okay. 16 So, if that is referring to that --16 see if it is convincing for you. So if you go to 17 no, I'm sorry, what you just read refers to 17 column 2, line -- I'm going to give you a whole 18 18 106 blocks, yes, that's what he says he's doing. bunch of excerpts from this to see how things are 19 19 Although we don't know anything about what residual with this with respect if it is satisfactory. It 20 state is in there and how the other blocks perform 20 says --21 the vertex processes simultaneously. 21 Q. What line? 22 Q. I want to make sure we agree on what the 22 A. I was just going to get to that. Let's 23 23 paragraph says at column 10 starting at line 43 and read that top paragraph, "according to another 24 24 going to line 47. That paragraph is referring to aspect of the present invention, there is provided a Page 125 Page 123 processing going on within one processing block 106, 1 1 data -- there is provided a data processing 2 right? 2 apparatus comprising an array of processing elements 3 3 A. One slice of 106, yes. which are operable to process respective data items 4 Q. And that paragraph is saying that the 4 in accordance with a common received instruction. 5 processing elements within one processing block will 5 When the processing elements are operably divided 6 be doing either pixel operations or nothing. Do we 6 into plurality of processing blocks having at least 7 7 agree? one processing element and the processing blocks 8 A. Yes, that's what it says. But I 8 being operable to process respective group of data 9 9 qualified my response -- my response to you that items, this data items refers to a general group of 10 those other blocks, fusion blocks, are doing other 10 data that is related to graphic processing including 11 11 things including vertex operations. vertices and pixels." 12 12 So let's continue with that. Same Q. Okay. Where does it say that? 13 A. We are going to find out. 13 column. 14 14 Q. Look at it. Let me give you all of the evidence 15 15 Can I have a pen or something to mark the that I found and then you can go and attack it if 16 areas that I want to refer to later on? Can I mark 16 you wish. But let me finish the thought up to that 17 the exhibit? 17 point. Is that okay? 18 18 Q. Yes, please do. Q. Hm-hmm. 19 19 A. Okay. I didn't know what the rules were. A. Column 2, line 52 or 51, "however, this 20 Let me give you some explanations. If you are not 20 embodiment is purely exemplary. This goes to the 21 happy, then I'll continue reading. How's that? We 21 embodiment issue that we have talked about before. 22 22 call that prefetching. And it will be readily apparent that techniques and 23 23 Okay. So are you ready? architecture described here for processing graphical 24 Q. I am. At some point I am going to want 24 data are equally applicable to other data types such

Page 126 Page 128 1 as video data and so on." 1 things because they're different data types, it's a 2 2 If it covers video data, it for sure multi-threaded, and it has the capability to launch 3 3 covers vertex and pixel because it is inherent in these instructions to each slice of the PEs. 4 the discussion of graphics processing. So I would 4 So, with that respect and the fact 5 5 say this guy basically went beyond graphics. that he talks about embodiments that could do more, 6 6 So let's go to column 4, line 25. I consider that his invention or their invention is 7 "As will be explained in greater detail below the 7 consistent with the fact that they can do PE and -processing core 10 includes a number of control 8 8 they could do pixel and vertices simultaneously. 9 units, thread manager 102, a rate controller 104, 9 Q. When you say they could do pixel and 10 and channel controller 108 and bidding unit 1069 per 10 vertices simultaneously, you mean the processing block, and micro code store 105. These control 11 11 blocks, one processing block could do vertex and one 12 units control the operation of a number of 12 processing block could do pixel, that's your processing blocks which perform the graphic 13 13 position? 14 processing itself." 14 A. That's right. 15 15 Q. The section that you didn't cite is Again, vertex and pixel are a 16 divisible part of the graphic processing. So it is 16 column 18. 17 also there for somebody who is reading this as an 17 A. I didn't get to it. I thought people 18 18 expert or a person with ordinary skill in the art. were getting impatient. 19 19 Q. In column 18 as we talked about earlier Again, the same column, it says the 20 array of --20 says at line 19 -- I guess, starting at line 17, 21 Q. What line? 21 "this is implemented by creating multiple bin lists 22 A. Sorry. Line 58, "the array 1061 of 22 per region, one for every processing block 106 that 23 23 processing elements provide a single instruction is processing geometry data. This allows the 24 multiple data processing structure." That's a data 24 geometry output phase to proceed in block parallel Page 127 Page 129 1 block parallel that I mentioned to you before. It 1 mode." 2 is SIMD. 2 A. Hm-hmm. 3 3 Q. The block parallel mode means that all "Each PE in the array 1061 is 4 supplied with the same instruction, which is used to 4 the blocks 106 are doing geometry operations, right? 5 process data specific to the PE concerned." So 5 MR. PLUTA: Object to form. 6 basically they have all the instructions they need 6 BY THE WITNESS: 7 7 to perform the different data types that we just A. It's an SIMD, yes. I think if you 8 talked or I just talked about earlier. 8 recall, I read that these blocks operate in SIMD. 9 9 Now, notice that there is the thread BY MR. TUMINARO: 10 10 controller at the top of that page, right, if you Q. If the block parallel mode is referring 11 11 could look at it. It says a thread controller, and to blocks 106, right? 12 the job of that thread controller is basically to 12 A. Yes. Each slice can do an SIMD 13 dole out --13 operation. 14 14 Q. What page? Q. And in line 30 of that same column 18, it 15 15 A. Thread manager. Thread manager is 102, says, "a record is kept of how many primitives are 16 figure 3 at the very top. Okay. 16 written to each bin so that regions can be sorted 17 So the thread manager for a 17 into similar size groups for block parallel 18 18 rasterization." multi-threaded processor, which is responsible for 19 19 different data types, is equally responsible for Do you see that? So block parallel 20 launching vertices. And the way it's shown in its 20 rasterization means that all the blocks 106 are 21 design, it could easily provide that to these PEs 21 doing rasterization, right? 22 22 simultaneously. But not to one group of PEs because A. One slice.

23

24

23

24

they would be doing SIMD for pixels, as you said,

and I also agree. But they could be doing different

Q. Block parallel means only one slice?

A. Again, if you go back to -- so this

Page 130 Page 132

- is parallelism -- the only way I can explain it to
- 2 you is -- why don't we just use books as these
- 3 fusion blocks. I have ten books on a book shelf.
- 4 One book is doing rasterization. The other book is
- 5 doing vertex operations. The other book may be
- 6 doing something else. The design is not preventing
- 7 you from designing it that way and the inventors
- 8 talk about it, the flexibility of the design.

So I agree with you that one slice or one book of that bookshelf is doing what you said but the other books are doing different things and they can do different things because the thread is distributing information to different blocks.

Q. What's your interpretation of Stuttard's disclosure of block parallel rasterization? Doesn't that mean that all the processing blocks 106 are doing rasterization, they're doing that in parallel?

MR. PLUTA: Object to form.

19 BY THE WITNESS:

1

9

10

11

12

13

14

15

16

17

18

6

7

8

9

10

11

12

13

14

15

16

17

18

- 20 A. One slice is doing the same thing. The 21 other slice is doing different things.
- 22 BY MR. TUMINARO:
- 23 Q. What is the parallel -- what does

24 parallel mean in that?

MR. PLUTA: Objection to form. BY THE WITNESS:

3 A. You have your hands like this so I have a 4 feeling that you want to say that all the fusion

5 blocks you're talking about. I disagree with that.

6 One is doing that. The other slice might be doing 7 something else. He has that flexibility.

BY MR. TUMINARO:

1

2

8

9

10

11

12

13

14

19

20

21

22

23

24

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

Q. So it's your interpretation that when he says block parallel, the block that he's talking about there is not processing block 106; is that your --

MR. PLUTA: Object to form. BY THE WITNESS:

15 A. No. It is processing block 106. Let 16 help refer you to the figure if you don't mind. For 17 the record, we're looking at figure 3.

18 BY MR. TUMINARO:

Q. Mark it up.

A. Okay. I'm going to letter name them. So it will be A, B, and C. So I have shown you -- of course, there's a dot dot, here, right? It could be like E, G, F. So that is one slice. Okay.

A. That's right because you got the PEs.

They can work in SIMD fashion. There is another

A. That's right. It could do the SIMD, and

work on different data types. The slice D and so

on. So it gives you the capability to function

Q. Okay. Referring to figure 3 -- thank

you. Referring to figure 3, block 106 on the top,

which you labeled as A, is shown as being parallel

A. Yeah. I see where you're -- Sorry, you

about parallel there. We're talking about parallel

might be misunderstanding here. We're not talking

multiple SIMDs at the same time.

it could be doing something else. He says that. We

Page 133

Q. A is one slice?

slice right behind it.

Q. Slice B?

to block B, correct?

Page 131

1 A. SIMD, single instruction multiple data.

3 Morphosys is an SIMD. If you read the paper, you

4

5

Q. You don't interpret block parallel to mean that all the blocks are doing the same parallel

A. That's what SIMD is, single instruction multiple data, but you have multiple slices. Think of it as a stack of these guys. Each slice -- so, if I wanted to do the morpho this way, I would have had two layers of morpho. One would be doing one type and the other another type. That is consistent with what he says.

Q. Maybe we're talking past each other. The slices that you're talking about in Stuttard -strike that.

19 The slices that you're talking about 20 Stuttard calls processing block 106, right?

21 A. Yes.

22 Q. So, when he says block parallel, doesn't 23 he mean that the processing blocks 106 are doing 24 parallel operations?

2 The best way to explain SIMD -- morpho was an SIMD.

saw I mentioned that it was SIMD. SIMD is single

instruction multiple data.

type of operation?

19 A. I understand what -- I see what your 20 confusion may be.

here.

21 22 23

24

Q. Your interpreting block parallel -- when Stuttard is talking about block parallel in column

Q. So -- but just to be clear --

18, he's not referring to processing block 106? A. No, no. Not by any stretch. If he's

34 (Pages 130 to 133)

Page 134 Page 136 1 inventing his own definition, that's a different about SIMD, yeah. 1 2 ball game. Somebody who is working in the field, a 2 BY MR. TUMINARO: POSITA, would understand a block parallel, it means 3 Q. Column 18 in the lines that we read that PEs here are working in parallel. They're 4 doesn't say SIMD? 5 5 doing an SIMD function. So just think of it as 106 A. It is implied. 6 slice A is a morpho. 6 Q. Doesn't say it, right? 7 Q. How would a person of ordinary skill in 7 A. A POSITA would know. 8 the art know that when Studdard talks about block 8 Q. Just SIMD does not appear in column 18? 9 parallel mode in column 18 -- let me start over. 9 MR. PLUTA: Objection, form. 10 How would a person of ordinary skill 10 BY THE WITNESS: 11 in the art know in 2003, reading Stuttard, that 11 A. Out of 33 pages, you are referring to a 12 Stuttard meant block parallel mode in column 18 to 12 paragraph that doesn't have SIMD, but he talks about 13 refer to an SIMD device? 13 SIMD. It is clear to a POSITA that it is SIMD. So 14 A. Did we not say what degree the person 14 you can take whatever you want from that paragraph. 15 should have as a POSITA? I think we did opine that 15 I disagree with that. 16 he has to have a degree in electrical engineering or 16 BY MR. TUMINARO: 17 computer science, and that topic is covered. And 17 Q. Switching gears --18 18 somebody working in the field with a background, A. My resume. 19 somebody taking course in engineering, especially 19 Q. Earlier we talked about DX10 and open GL or the current version of DX and the current version 20 computer engineering and computer science would know 20 21 that what SIMD is. We cover it. Our degree is a 21 of open GL and we agreed that they require a unified 22 very typical ABET accredited. I think that's true 22 shader? 23 23 for everybody. MR. PLUTA: Objection, form. Objection, 2.4 Q. SIMD was well known in 2003? 24 relevance. Page 137 Page 135 1 BY THE WITNESS: A. Morpho was an SIMD, so ... 2 Q. In fact, at other times when Stuttard 2 A. From the document, you provided and I 3 meant SIMD, he explicitly said SIMD, right? 3 read, that's what it says. 4 A. That's right. 4 BY MR. TUMINARO: 5 5 Q. For example, in column 1, line 21, it Q. And do you also remember that now? 6 says, "the present invention relates to parallel 6 MR. PLUTA: Objection, form. Objection, 7 7 data processing apparatus, in particular to SIMD, relevance. 8 (single instruction multiple data) processing 8 BY THE WITNESS: 9 9 apparatus." Right? I believe that at the time we were 10 10 A. Yes. looking at the DX and open GL, they were referring, 11 11 MR. PLUTA: Objection, form. but I don't know the exact year of the open GL we 12 12 looked at. They transitioned to this as you can BY MR. TUMINARO: 13 O. So when Stuttard meant SIMD, he said 13 see. So I cannot say for certain that when I looked 14 14 SIMD, right? at open GL for another thing. 15 15 MR. PLUTA: Object to form. BY MR. TUMINARO: 16 BY THE WITNESS: 16 Q. Okay. So a GPU today, as of today, to be 17 A. When he said SIMD, he meant SIMD. I 17 fully compatible with the current version of DX and 18 18 the current version of open GL, that GPU would have don't know. I mean, that's probably what he meant. 19 BY MR. TUMINARO: 19 to provide a unified shader, right? 20 20 MR. PLUTA: Objection, form. Objection, Q. So when he wanted to refer to a single 21 instruction multiple data, he said SIMD, right? 21 relevance. 22 22 MR. PLUTA: Objection, form. BY THE WITNESS: 23 23 A. I could not tell you that for sure. Can BY THE WITNESS: 24 A. He says a previously proposed -- he talks 24 there be a design around where you do it without a

Page 138 Page 140 1 unified shader? Anything is possible. that. I cannot answer. Sorry. 1 2 2 BY MR. TUMINARO: Q. Well, I'm asking to be fully compliant 3 3 Q. You don't know one way or the other if with those APIs, the GPU would have to provide a 4 unified shader, to be fully compliant? 4 you provided an opinion on objective indicia of 5 5 MR. PLUTA: Objection, form. nonobviousness? 6 6 MR. TUMINARO: Not a design around. MR. PLUTA: Objection, form. 7 MR. PLUTA: Objection. Objection, form. 7 BY THE WITNESS: 8 8 A. I would be glad to look at my report. I Objection, relevance. 9 9 BY THE WITNESS: have prior art. I have other items that I've 10 10 A. It's hard to say, Counsel. It's hard to discussed. 11 11 BY MR. TUMINARO: sav. 12 BY MR. TUMINARO: 12 Q. Take a look at your report. Tell me. 13 O. Why is it hard to say? 13 Did you provide an opinion on objective indicia of 14 14 A. Because you could design it that it is nonobviousness? 15 interacting with the APIs the DX10 and 11 and open 15 A. I don't believe I have. 16 GL and still not completely a unified shader. It's 16 MR. PLUTA: We've been going about an 17 hard to say. You have to design it. I cannot 17 hour and a half. Is this a good breaking point? 18 18 MR. TUMINARO: Sure. We can take a answer that the way you are asking me. 19 19 Q. What do you mean by completely a unified break. 20 shader? What is the qualifier "completely" mean? 20 THE VIDEOGRAPHER: We're going off 21 MR. PLUTA: Objection, form. Objection, 21 record. The time is 1:49 p.m. 22 relevance. 22 (Whereupon, a short break in the 23 23 BY THE WITNESS: proceedings was taken.) THE VIDEOGRAPHER: We're back on record. 2.4 A. A unified shader has to be a fully 2.4 Page 139 Page 141 programmable solution. There might be ways of doing 1 This is the beginning of Media 3. The time is 2 this without adhering to that objective. Yeah. 2 2:12 p.m. 3 3 That's basically what I have to say. I cannot BY MR. TUMINARO: 4 answer the question without more analysis. 4 Q. Would you turn with me to Lindholm 5 BY MR. TUMINARO: 5 reference. I forget what exhibit number it is. 6 Q. I guess what I'm hung up about, if the 6 MR. PLUTA: 6. 7 7 GPU has to satisfy everything that's in those APIs, MR. TUMINARO: Thank you. 8 and one of things that's in the API is a unified 8 BY MR. TUMINARO: 9 shader, how is it that the GPU could not provide a 9 Q. Would you circle what you're calling the 10 10 unified shader? arbiter? What in Lindholm corresponds to the 11 MR. PLUTA: Object, form. Objection, 11 arbiter? 12 12 relevance. A. Which figure? 13 BY THE WITNESS: 13 Q. Whatever figure you think best depicts 14 14 A. This is outside the scope of what I the arbiter. 15 15 MR. PLUTA: Object to the form. worked on. So I will not be able to give you a very 16 accurate and professional answer. It requires more 16 BY MR. TUMINARO: 17 work. I would be glad to work on that and get back 17 Q. Would you write arbiter next to it? 18 18 to you. A. (Witness marking document.) 19 BY MR. TUMINARO: 19 Q. Would you circle for me what you're 20 Q. Because you didn't provide any opinion on 20 saying corresponds to the claim claimed plurality of 21 objective indicia of nonobviousness? 21 command processing engines from the '053 patent --22 22 MR. PLUTA: Objection, form. that was a terrible question. Let me start again. 23 BY THE WITNESS: 23 Would you please look at the Lindholm 24 A. Now you're wrapping a legal issue on 24 reference and circle in Lindholm what you say

	Page 142		Page 144
1	corresponds to the claimed plurality of command	1	reserved and managed according to an alternative
2	processing engines?	2	memory space reservations process. Figure 5 is
3	A. (Witness marking document.)	3	described in relation to figures 1 and 2. Memory
4	Q. Just to be clear, what figure are you	4	resource 200 of figure 5 contains each element shown
5	marking up?	5	in figure 2. Memory resource also contains a second
6	A. Figure 4.	6	memory section 520 having at least two memory spaces
7	Q. Figure 4 of Lindholm?	7	505 for use by threads of second set of threads
8	A. Yes. I think so, yes.	8	executing on graphics processor 125. All threads of
9	Q. Okay. Would you turn with me to the	9	the first set of threads are of the first thread
10	Morton reference, which is Exhibit 4?	10	type and are each reserved in memory space 205.
11	A. Are we done with this?	11	So basically memory space 205 is used
12	Q. You can put it aside.	12	for the first set of threads and then on memory
13	A. Yes.	13	space 505 is for the other type. So, this is
14	Q. Sorry. I didn't realize you were there.	14	referring to this passage. Memory space 505 having
15	I apologize.	15	a second memory space second thread type being
16	Would you circle for me in Morton	16	different than the first thread type.
17	what corresponds to the claimed plurality of command	17	BY MR. TUMINARO:
18	processing engines?	18	Q. Is memory space 505 on figure 5?
19	MR. PLUTA: Objection, form.	19	A. That's what it's referring to in this
20	BY THE WITNESS:	20	section of the document.
21	A. He talks about this being one slice or	21	O. Is that one of the is that the first
22	multiple slices, but at least figure 1 shows one	22	portion or the second portion?
23	slice. I'm going to make the assumption that so	23	A. Second memory thread type.
24	as I pointed out, it's 140 through 170. These are	24	Q. Would you circle that then?
	Page 143		Page 145
1	the graphics processors.		A. It's done.
2	BY MR. TUMINARO:	2	Q. Would you label it?
3	Q. Would you write next to that command	3	A. (Witness marking document.)
4	processing engines.	4	Q. Thank you.
5	A. (Witness marking document.)	5	It was figure 5 that you marked up,
6	Q. Just so we're clear, you're marking	6	just so the record is clear?
7	figure 1 of Morton?	7	A. Yeah.
8	A. Yeah.	8	Q. Would you turn with me to Paragraph 131
9	Q. Claim 1 of the '053 patent recites a	9	of your declaration. I'll read for the record with
10	first portion and a second portion for the memory	10	respect to claim 7. "Because Stuttard also
11	device.	11	discloses that the processing elements 1061 perform
12	A. Yes.	12	'lighting and shading' functions, the processing
13	Q. Would you circle the first portion and	13	elements 1061 must necessarily include a texture
14	the second portion that you say strike that.	14	processing engine."
15	Would you circle in Morton what you	15	Do you see that?
16	say corresponds to the claimed first portion and	16	A. Yes.
17	second portion from the '053 patent?	17	Q. Is it your position that lighting
18	MR. PLUTA: Objection, form.	18	functions necessarily require texture processing?
19	BY THE WITNESS:	19	MR. PLUTA: Objection, form.
20	A. So referring to column 15, 48 through 67.	20	BY THE WITNESS:
21	Also referring to the claim construction page	21	A. It speaks for itself. Lighting and
22	chart sorry my claim construction chart,	22	shading functions must necessarily include the
23	figure 5 shows a conceptual diagram of a memory	23	texture processing engine. I cannot add anything
د ک	resource 200 containing memory spaces that are	24	more than what is in here.
24			

Page 146 Page 148 1 BY MR. TUMINARO: BY MR. TUMINARO: 1 2 2 Q. Isn't it true, though, that a shading Q. Can you answer my question? 3 3 MR. PLUTA: Objection, form. Objection, function can be performed without texture 4 4 relevance. Asked and answered. processing? 5 5 A. In my opinion, predominantly, this is how BY THE WITNESS: 6 it is done. 6 A. I did the best I could. 7 7 (WHEREUPON, document marked as Q. Predominantly, but it is possible that a 8 shading function can be performed without texture 8 Exhibit No. 10.) 9 9 processing, right? MR. PLUTA: I'm going to object to the 10 10 relevance of this, the introduction of this exhibit A. No. In my opinion, that's the way it's 11 done. 11 into evidence. 12 Q. So shading functions have to include 12 BY MR. TUMINARO: 13 texture processing? 13 Q. This is Exhibit 10. You have been handed 14 MR. PLUTA: Objection, form. 14 what has been marked as Exhibit 10 for 15 15 identification purposes. This paper is titled BY THE WITNESS: 16 A. It says, must necessarily include the 16 Design and Implementation of a Rendering Algorithm 17 texture processing engine. So that's clear what it 17 in a SIMD Reconfigurable Architecture (Morphosys). 18 18 says: Must necessarily. Do you see that? 19 19 BY MR. TUMINARO: A. Yes. 20 Q. Is it possible that a shading function 20 Q. In the listing of inventors there is one 21 can be performed, and in doing that shading 21 Nader Bagherzadeh. Do you see that? 22 function, there's no texture processing? Is that 22 MR. PLUTA: Objection, form. 23 23 BY THE WITNESS: possible? 24 2.4 A. Not correct. We're -- this is an MR. PLUTA: Objection, form. Page 147 Page 149 1 BY THE WITNESS: article. It's not a patent. So it's not an 2 A. It is my opinion that a texture 2 invention. 3 processing engine must include lighting and shading. 3 BY MR. TUMINARO: 4 BY MR. TUMINARO: 4 Q. Sorry. I said inventors? Sorry. 5 Q. I can read your declaration. My question 5 A. We don't consider ourselves inventors. Q. One of the authors listed on Exhibit 10 6 is a little bit different. Is it possible that a 6 7 shading function can be performed; and in doing that 7 is an individual named Nader Bagherzadeh? 8 shading function, there's no texture processing? 8 A. Yes. 9 MR. PLUTA: Objection, form. Objection, 9 Is that you? O. 10 10 relevance. A. Yes. 11 BY THE WITNESS: 11 Q. You're one of the authors on this paper? 12 12 A. I mean, somebody could design something, A. Yes. 13 but it is my opinion that that should include 13 O. If you would turn with me to the third 14 14 lighting and shading. page. It is not numbered. The third page of text. 15 BY MR. TUMINARO: 15 On the right-hand side of the page, 16 Q. Is it possible that a lighting function 16 right hand column, there's a formula, which is the 17 can be performed that does not involve any texture 17 first line on the right-hand column. It says Id 18 18 processing? equals I and it goes on. Do you see that? 19 19 MR. PLUTA: Objection, form. Objection, A. Yes. 20 20 Q. If you look to the left-hand column in relevance. 21 BY THE WITNESS: 21 the last full paragraph, it starts with "in order to 22 22 A. It is my opinion that lighting and obtain." 23 23 shading should be part of the texture processing. Do you see that? 24 24 A. Yes.

Page 150 Page 152 1 Q. I'll read it. "In order to obtain an 1 mapping, right? 2 2 image with enough realism, we apply to each pixel a MR. PLUTA: Objection, form. Objection, 3 Gouraud shading algorithm." Did I say that 3 relevance. 4 4 BY THE WITNESS: 5 5 A. Gouraud shading. A. The function that is responsible for 6 Q. Can you explain what Gouraud shading is? 6 lighting and shading would be called texture 7 MR. PLUTA: Objection, form. Objection, 7 processing engine, and this would be part of that. 8 8 relevance. But we did not call it that much because this was a 9 BY THE WITNESS: 9 limited time. But an expanded version of this 10 10 A. It is says we calculate the light should have a texture processing engine named and 11 intensity for each triangle, basically like 11 included this function. 12 perpendicular to the surface of a vertex to that 12 BY MR. TUMINARO: 13 other primitive. 13 Q. Okay. There might be an expanded 14 BY MR. TUMINARO: 14 version, but yes or no, this shading function that's 15 Q. A Gouraud shading, does that include 15 disclosed in your own paper is an example of a 16 16 shading function that does not include texture texture mapping? 17 MR. PLUTA: Objection, form. Objection, 17 mapping? 18 18 MR. PLUTA: Objection, form and relevance. 19 19 BY THE WITNESS: relevance. 20 A. It could. But in this case, we didn't 20 BY THE WITNESS: 21 talk about it. 21 A. This function is part of a texture 22 BY MR. TUMINARO: 22 processing engine, the box responsible for that. 23 23 Q. In this case, does it include texture BY MR. TUMINARO: 24 mapping? 24 Q. Is it your testimony that this Gouraud Page 151 Page 153 1 MR. PLUTA: Objection, form. Objection, 1 shading that's described in your paper necessarily 2 2 occurs in a texture processing engine? relevance. 3 3 BY THE WITNESS: MR. PLUTA: Objection, form. Objection, 4 A. Yeah, in this case, we did not talk about 4 relevance. 5 5 texture mapping, but we could have. It was just not BY THE WITNESS: 6 part of the design. It would have been appropriate 6 A. It belongs necessarily to a texture 7 7 to add that as well. processing engine, yes. 8 BY MR. TUMINARO: 8 BY MR. TUMINARO: 9 9 Q. But it is a fact that in this shading Q. Is that how it was implemented in the 10 10 algorithm there is no texting mapping? example that you described in your paper? 11 MR. PLUTA: Objection, form. Objection, 11 MR. PLUTA: Same objections. 12 12 BY THE WITNESS: relevance. 13 BY THE WITNESS: 13 A. I think we basically simulated a lot of 14 14 A. What it is is that it gives you an stuff here. We did not implement this as far as I 15 15 opportunity to find Gouraud shading, but a box that remember. This was just a paper analysis. 16 does texture -- what did we call it? A texture 16 BY MR. TUMINARO: 17 processing engine would include this particular 17 Q. Did this paper analysis include the 18 18 simulation of a texture processing engine? function as well. So -- so, this is part of the 19 19 MR. PLUTA: Objection, form. Objection, texture mapping engine. 20 20 BY MR. TUMINARO: relevance. 21 Q. This is your paper and you talk about 21 BY THE WITNESS: 22 22 Gouraud shading, right? And this Gouraud shading A. It is assumed that that is part of the 23 23 texture processing engine, yes. that you talk about in your paper, it is an example MR. TUMINARO: We have no more questions 24 where a shading function does not include texture 24

Page 154 Page 156 at this time. 1 1 A. Instruction cache, which is Box 410. 2 2 MR. PLUTA: Let's go off the record. Q. Let's look at another document, the 3 THE VIDEOGRAPHER: Going off record. The 3 petition, which is Exhibit 9. Do you recall Counsel 4 took you to Page 13 I believe of the --5 5 (Whereupon, a short break in the A. Okay. 6 proceedings was taken.) 6 Q. -- petition? Do you recall that counsel 7 THE VIDEOGRAPHER: We're back on record. 7 asked you about Page 13 and regarding the thread 8 The time is 3:06 p.m. 8 control buffer 420? 9 EXAMINATION 9 A. Correct. 10 10 Q. And you recall counsel asked you whether BY MR. PLUTA: 11 Q. Welcome back. I just have a couple of 11 you agree with petitioner's statement and you 12 questions for you. 12 said -- and I'm paraphrasing here -- you agree with 13 Do you recall the discussion about 13 it because it includes the thread control buffer 14 VHDL and in specifically in relation to Exhibit 3 --14 420, right? 15 15 A. Yes. MR. TUMINARO: Objection, leading. 16 Q. -- your paper? 16 BY MR. PLUTA: 17 A. Yes. 17 Q. Do you recall your testimony regarding 18 18 Q. Now, putting aside my objections to the Page 13 of the petition? 19 paper, I have a couple of questions for you on that 19 A. Yes. 20 topic. You recall the questions counsel asked you 20 Q. And counsel asked you about the position 21 about building a chip, correct? 21 with respect to I believe it's the first paragraph 22 A. Yes. 22 of Page 13? 23 Q. You also mentioned, I'm paraphrasing your 23 A. Okay. 24 testimony here, that to you building a chip means 2.4 And you agreed that the memory device Page 157 Page 155 1 fabrication of a chip, correct? includes the thread control buffer, correct? 2 A. Right. 2 A. Correct. Q. And you just testified that you also 3 Q. You also mentioned a process of massaging 3 4 during the process of designing a computer chip in 4 believed that it also includes the instruction 5 particular after simulation of it? 5 cache, correct? 6 A. Yes. 6 A. Correct. Q. I'd like to turn to Page 22 of the 7 Q. And is it true that you testified that 7 there's a lot of trial and error involved in the petition and direct your attention to the second 9 9 design of a computer chip? full paragraph on that page. Could you read that 10 10 A. Yes. second full paragraph, please, for the record? 11 11 A. "As discussed above, Lindholm discloses Q. I also like to direct you now to 12 12 Lindholm, which is Exhibit 6, just for reference, instruction cache 410 and thread control buffer 420 13 and then I'm also going to take you to Paragraph 72 13 which corresponds to the claim at least one memory 14 14 of your declaration. device." 15 15 A. Yep. Q. Now, after reading that, is it your 16 Q. Do you recall questions about the thread 16 position that the thread control buffer in 17 control buffer 420? 17 combination with the instruction cache corresponds 18 18 A. Yes. to the claimed memory device consistent with the 19 19 Q. Is your opinion that the thread control petitioner's petition here in this case? 20 buffer is the only element that corresponds to the 20 A. Yep. This is exactly what I had in mind. 21 claimed memory device? 21 Q. Let's switch one more topic. 22 22 No. A. Okay. A. 23 23 Q. What else are you pointing to as Q. Do you recall the questioning regarding 24 corresponding to the claimed memory device? 24 the ALU resource and texture fetch resource

	Page 158		Page 160
1	divisions with respect to the background of the '053	1	REPORTER CERTIFICATE
2	patent?	2	
3	A. Yes.	3	I, JO ANN LOSOYA, a Certified Shorthand
4	Q. The command threads in the ALU resource	4	Reporter within and for the County of Cook and State
5	division and texture fetch resource divisions are	5	of Illinois, do hereby certify:
6	different types of threads, correct?	6	That previous to the commencement
7	A. Correct.	7	of the examination of the witness, the witness was
8	MR. TUMINARO: Objection, leading.	8	duly sworn to testify the whole truth concerning the
9	BY MR. PLUTA:	9	matters herein;
10	Q. Are the command threads in the ALU	10	That the foregoing deposition
11	resource division and the texture fetch resource	11	transcript was reported stenographically by me, was
12	division different types of threads?	12	thereafter reduced to typewriting under my personal
13	A. They are.	13	direction and constitutes a true record of the
14	Q. Are pixel command threads and vertex	14	testimony given and the proceedings had;
15	command threads also different types of threads?	15	That the said deposition was taken
16	A. Yes.	16	before me at the time and place specified;
17	MR. PLUTA: No further questions at this	17	That I am not a relative or
18	time.	18	employee or attorney or counsel, nor a relative or
19	EXAMINATION	19	employee of such attorney or counsel for any of the
20	BY MR. TUMINARO:	20	parties hereto, nor interested directly or
21	Q. I have questions. Did you talk with your	21	indirectly in the outcome of this action.
22	counsel about the substance of your testimony after	22	
23	I said I had no further questions at this time?	23	
24	A. Absolutely not. They did not even say	24	
	Page 159		Page 161
1	hello to me, which I was disappointed.	1	IN WITNESS WHEREOF, I do hereunto set my
2	MR. TUMINARO: I have nothing else.	2	hand this August 15, 2015.
3	THE VIDEOGRAPHER: This concludes the	3	nata dila Magdat 15, 2015.
4	videotaped deposition of Nader Bagherzadeh. The	4	
5	time is 3:12 p.m. We're off record.	5	
6	(Witness excused at 3:12 p.m.)	6	
7	(Witness excused at 3.12 p.m.)	7	
8		8	
9			JO ANN LOSOYA, CSR
10		9	C.S.R. No. 84-002437
11		10	
12		11	
13		12	
14		13	
15		14	
16		15	
17		16	
18		17	
19		18	
20		19	
21		20	
22		21	
		23	
23		23	

				1 490 107
A	130:9 156:11,12	API 90:24 92:20,22	25:4,10,18 107:2	attack 125:15
	agreed 120:7,12,16	139:8	107:5	attempts 102:17
AAPA 50:8 52:2,4	136:21 156:24	APIs 103:23 104:8	architecture 12:20	attention 68:13
ABET 134:22	agreeing 82:12	138:3,15 139:7	13:9 19:12 20:17	157:8
abilities 56:16	ahead 72:24	Apologies 83:19	25:12 26:17,18	attorney 160:18,19
able 38:12 39:7	algorithm 148:16	apologize 62:11	55:17 61:3 67:17	attorneys 5:13
55:17 61:11 91:13	150:3 151:10	142:15	68:3,4 86:6 93:15	AT&T 15:15
92:19,20,22	alive 46:22	apparatus 125:2	94:3 103:22	August 1:20 161:2
104:21 139:15	allow 61:9,10	135:7,9	107:13 110:10,15	authenticity 96:19
Absolutely 7:19 8:2	allows 74:18	apparent 125:22	125:23 148:17	96:20
8:4 16:22 85:9 158:24	128:23	Appeal 1:4 5:10	architectures 27:15	authors 149:6,11
abstract 67:5	alluded 45:21	appear 136:8	27:17,18	available 13:10
	alluding 107:19	Appeared 2:6,11	areas 123:16	59:17,22 60:5,8
academia 16:5,10 16:12	alpha 54:16 55:11	2:16	array 70:16 125:2	60:11,18,22,24
	55:13,20 56:10	appears 13:15	126:20,22 127:3	61:5,7,16,20,21
accredited 134:22	58:13	41:17 63:7	art 26:14 27:6,11	61:23,24 62:4,7
accurate 16:6,7 23:15 47:22	alternative 144:1	appendix 8:24	51:5,23 91:11	62:17,21 63:5,6
	ALU 52:9,14,21	applicable 125:24	100:18,21,23	65:6 111:6,11
106:12,15,16 109:14,18,18	53:3,12,20 54:3	application 29:18	103:6 115:20	112:7,19,23
111:13 115:23	54:11,18 56:4,8	34:17,23 35:2	116:10,23 117:16	117:17,17
139:16	56:11,21 57:4,13	90:19 92:19 110:8	117:16 126:18	Avenue 2:9,13
	57:23 58:8 157:24	110:8 117:5	134:8,11 140:9	aware 90:2
action 160:21	158:4,10	applied 100:24	article 149:1	a.m 1:19 5:4 39:21
activities 76:13	ALUs 55:3	applies 63:2	aside 142:12	40:1 85:17
119:9,11 add 9:8 15:9 56:18	amount 57:12	apply 76:15 150:2	154:18	
58:5 65:22 67:11	analysis 75:12	appreciate 12:12	asked 10:8 17:9,23	B
	99:18 101:18	46:11 49:24 82:21	23:24 26:5 30:16	B 4:9 132:21 133:4
89:22 102:1 145:23 151:7	102:11,16 139:4	95:23	35:6,24 58:2	133:13
addition 96:20	153:15,17	appropriate 110:15	67:15 148:4	back 23:7 26:9 29:8
additional 96:12	analyze 100:6,9	110:20 151:6	154:20 156:7,10	30:11 31:7 39:24
address 6:14,15,17	analyzed 99:21	approximately 5:4	156:20	40:3 41:14 53:1
30:7 55:17	Ann 1:14 2:19,23	arbiter 37:20 38:1	asking 17:13 25:3,4	58:11 68:9 73:20
addressed 32:8	5:2 160:3 161:8	38:4 58:23,24	34:19 38:16 58:12	85:20,24 105:11
addressed 32.8 adhering 139:2	announce 5:14	59:7,12,18 61:5,6	67:5,16 87:12	105:14 115:14
Adjusted 110:17	answer 7:11,14,21	61:6,8,15 62:9,20	89:16 90:23	124:1,2 129:24
admitted 51:23	12:12,13 23:19	62:24 63:3,18,18	105:23 138:2,18	139:17 140:24
advertise 92:21	25:17 30:21 35:15	64:19 65:10,11,14	aspect 124:24	154:7,11
agencies 107:18	57:16 61:11 67:22	65:15 66:10,13,14	aspects 30:24	background 30:6
agency 14:9	92:20 97:11 102:3	67:2,6,7,8,10,12	asserting 75:3	31:8,9 32:12
agency 14.9 ago 45:22	138:18 139:4,16	69:17 86:24 88:10	assessment 104:11	52:13 110:1
agree 42:24 44:5	140:1 148:2	111:6,10,10,14,23	104:14	134:18 158:1
47:24 48:15,22	answered 26:5	112:7,8,9,12,18	assigned 119:8,9,10	Bagherzadeh 1:13
49:14 50:11 54:2	30:17 54:9 55:8	141:10,11,14,17	associated 54:17	4:3 5:12 6:6,13
54:10 62:18 64:3	56:15 58:3 148:4	arbitrate 65:16	assume 7:12 33:11	8:21 14:16 40:4
78:15 81:13 85:3	anticipate 50:12	arbitration 65:13	115:18 117:4	148:21 149:7
88:12,14 89:11	anticipates 50:3,4,9	architect 54:22	assumed 153:22	159:4
115:5 116:1	anticipating 50:10	architectural 20:24	assuming 117:12	Bagherzadeh's
120:14 122:22	anybody 102:4	21:14,21,23 23:9	assumption 142:23	9:20
123:7 127:24	117:4	23:9,21 24:1,7,8	ATI 1:10 5:9,17,20	balancing 94:8
123.7 127.21				
	•	•	•	•

	l	l	l	
ball 134:2	58:14	breaks 7:16	calling 141:9	chute 60:7,7
based 21:20 26:23	block 34:2 48:8	broad 64:17 87:9	calls 131:20	circle 141:9,19,24
26:24 104:21	69:9,10,10,11	Brown 2:2 5:6,22	capabilities 105:17	142:16 143:13,15
110:8	70:12 73:21,23,24	5:24	105:19	144:24
basically 64:23	74:3,17,19,23	Bryan 2:13 5:19	capability 106:9	circuit 25:24
109:7 111:2 112:8	75:5,15,23 76:4,7	buffer 41:4,11	128:2 133:8	citation 119:13
126:5 127:6,12	76:12,17 77:1,4	46:18 47:1,16	caption 5:8	cite 31:13,19 32:17
139:3 144:11	77:14,20 78:1,10	48:2 52:8,9,20,21	carried 80:17 120:1	53:15 128:15
150:11 153:13	79:5 80:22 81:24	52:22 64:9 113:5	case 5:8,9,10 8:11	cited 12:18,19,22
basis 80:11 110:13	82:1,16,17,23,24	113:9 114:6,10,15	8:13 10:10 35:3	cites 37:24
118:10	83:1,5,6,24 84:1	155:17,20 156:8	60:15 62:16 67:16	Civil 1:15
bathtub 33:13,14	85:1 120:3,8,24	156:13 157:1,12	69:18 87:11	claim 37:11,14
33:15	121:2,4,5,12	157:16	100:17 105:24	38:23 41:17,18
Beach 6:16	122:2,3,4,12	build 16:21 17:15	150:20,23 151:4	42:2,6,11 43:8,22
beginning 25:9	123:1,5 124:10	17:17 18:8	157:19	44:21 45:2 46:16
85:21 141:1	126:11 127:1	building 17:4,7,19	category 53:12	49:4,7,8 50:3,5,12
behalf 2:6,11,16	128:11,12,22,24	154:21,24	56:21	53:1,2 58:21
5:17,20,23 6:1	129:3,10,17,19,23	built 16:16 17:12	caveat 81:8	61:18,21,24 62:13
behavior 23:17	130:15 131:6,20	bullet 32:21 33:4	certain 12:4 30:24	63:7,13 86:16,20
24:20	131:22 132:10,10	94:7	51:24 54:23 87:21	87:13 88:6,10
behavioral 22:18	132:11,15 133:11	bunch 77:18 92:9	109:19 137:13	91:17 111:19
believe 26:13	133:13,21,22,23	124:18	CERTIFICATE	113:10 118:22
105:15 120:12	134:3,8,12	bus 66:1,4,10,13,15	160:1	141:20 143:9,21
137:9 140:15	blocks 70:15,22	66:16	Certified 160:3	143:22 145:10
156:4,21	71:3,12,16,19	business 75:24	certify 160:5	157:13
believed 157:4	74:4,6 75:7 76:8		change 110:11,20	claimed 41:5,18
Bell 15:15,24 16:2	77:10,15 94:12	<u>C</u>	changes 10:2 15:13	44:15 45:7,19
belly 77:18	122:7,10,18,20	C 108:1,7 132:21	channel 126:10	46:2 59:7 62:23
belong 53:11 56:21	123:10,10 124:9	cache 33:4,5,8,9,11	chart 46:16 118:22	63:3,18 101:11,20
57:3	125:6,7 126:13	33:16,19,22 34:2	143:22,22	101:23 102:12,17
belonging 52:17	128:11 129:4,8,11	34:4,6,13,14,20	check 31:14,21	113:9 114:10
belongs 153:6	129:20 130:3,13	34:21,24 35:8,9	50:7 72:4 82:6	115:7 118:7
best 56:15 131:2	130:16 131:7,23	35:13,16,20,20	111:20 118:21	141:20 142:1,17
141:13 148:6	132:5	44:20 45:23 46:1	chemists 110:4	143:16 155:21,24
better 20:6 33:17	bmechell@robin	110:18,20,20	Chicago 1:19 2:4	157:18
62:14	2:15	113:13 115:8	5:7	claims 51:24 62:14
beyond 100:6 126:5	Board 1:4 5:10	118:5,11,16,19	chip 16:17,21 17:4	86:12 102:4
bidding 73:3	boards 16:1	119:2 156:1 157:5	17:7,7,12,20,21	clarify 21:11 31:23
126:10	book 32:18 130:3,4	157:12,17	17:22 18:4,6,8	46:10,11 52:20
big 34:1	130:4,5,10	calculate 150:10	20:18,22 21:1,5	62:13 72:2
bigger 35:16	books 130:2,3,11	calculating 119:11	21:15,21 23:22	clause 50:23
bin 73:24 74:16	bookshelf 130:10	California 6:16,18	26:3 58:10 92:11	clear 7:1,5 9:19
76:3 128:21	box 151:15 152:22	6:19	106:8 107:19	14:24 30:14,22
129:16	156:1	call 65:18 91:12	110:22 154:21,24	41:10 44:13 45:5
bins 73:24 74:14	break 7:17,22 36:4	123:22 151:16	155:1,4,9	46:4,5,24 50:1,11
bit 7:9 147:6	39:17,19,22 40:5	152:8	chips 18:7 31:1	51:4 57:16 61:16
blend 55:13	85:14,18 105:5,9	called 6:7 18:8	92:2,8,9 101:16	82:8 83:23 112:11
blending 54:16	140:19,22 154:5	28:23 33:13 34:1	choice 110:10	121:24 122:1,8
55:11,20 56:10	breaking 140:17	152:6	choosing 62:18	133:18 136:13

142:4 143:6 145:6	61:22,22 63:4,4	concern 80:16	113:5,9 114:6,10	122:6 138:10
146:17	111:16 112:2,2,3	concerned 119:24	114:15 119:8	154:20 156:3,6,10
close 109:18	118:6,12,18,18	127:5	126:8,11,12	156:20 158:22
closed 60:13,22	141:21 142:1,17	concerning 79:24	155:17,19 156:8	160:18,19
code 1:15 23:10	143:3 158:4,10,14	160:8	156:13 157:1,12	counter 69:22
106:19 108:1,8	158:15	concludes 159:3	157:16	County 160:4
126:11	commands 39:15	condition 62:9	controlled 41:11	couple 6:23 154:11
colleagues 18:22	39:16 53:20 54:17	confused 69:21	controller 126:9,10	154:19
column 20:11	56:24 57:5,20,24	107:4	127:10,11,12	coupled 37:19 38:4
22:12 37:11,11	86:22 87:1,1	confusion 133:20	convincing 84:22	58:24
41:17 70:19,24	commencement	consider 87:13	124:16	course 9:18 26:7
72:18 73:19 77:13	160:6	102:20 104:24	Cook 160:4	28:7,8 132:22
				-
78:2,15 79:23	commencing 1:19	116:2,14,22 117:9	copied 102:4	134:19
80:3,4 83:11	comment 63:1 95:2	117:13 128:6	copy 13:13	court 1:16 6:2
119:13,18 120:22	95:23 101:7 104:4	149:5	copying 89:7,9,11	cover 134:21
121:9 122:23	104:12	consideration	91:5 101:11,19,23	covered 134:17
124:17 125:13,19	commercial 35:19	88:23 99:22	core 70:21 71:2,2,4	covers 126:2,3
126:6,19 128:16	89:8	considered 10:19	71:4 77:6,7,10,15	CPU 29:13,15,21
128:19 129:14	common 54:14	11:19,22,24 12:10	126:8	29:22,23 30:1
133:22 134:9,12	125:4	12:14 13:4,15	correct 6:22 7:15	69:9,11,21
135:5 136:3,8	companies 101:16	30:12,15 31:2	10:20,21 16:14	create 102:17
143:20 149:16,17	company 27:22	36:22 37:8 40:24	24:1 31:10,13	creating 74:13,15
149:20	compatible 92:3,12	70:8 100:18 103:6	36:19,20,22,23	128:21
combination 44:19	94:17 103:23	103:10,12 113:24	37:7,9 40:12,22	crunching 28:21
45:22 48:12	104:7 137:17	116:8	41:1 50:13 52:6	CSR 161:8
111:15 112:1	completely 138:16	considering 116:13	52:12 53:5,14,15	current 14:22
113:13 157:17	138:19,20	consistent 115:2	70:9 71:20 72:9	98:23 136:20,20
combining 117:20	completes 114:19	128:7 131:14	79:18 82:5 87:2	137:17,18
come 12:7 26:21	compliant 138:2,4	157:18	89:5 102:13,14	Curriculum 14:16
32:7 65:21 110:5	comprises 119:2	constitutes 160:13	118:3 119:5	curve 33:13,14
comes 32:12	comprising 41:20	construction	133:13 148:24	CV 14:22 15:1,1,3
coming 106:10	42:20 125:2	111:19 143:21,22	154:21 155:1	15:9 19:4,5
110:2	computation 29:17	consultant 14:16	156:9 157:1,2,5,6	CVs 15:4
comma 114:5	55:16,18,24 57:3	15:1,1	158:6,7	cycle 106:12,15,16
command 37:15,19	57:10,23 58:8	consumption 34:12	correctly 47:20	109:14,18,18
37:21 38:6,7,8,11	60:4 76:14 109:16	54:24	48:20 73:8 74:21	cycles 109:15
38:13,13,24 39:1	121:21	contacted 14:5,8	80:20 84:21 150:4	C.S.R 1:14 161:9
39:2,5,8,13,13	computations 55:2	containing 143:24	corresponding	C57 19:9
41:21,23 42:3,3,7	55:3 85:9 120:18	contains 78:7,22	155:24	
42:8,15,16,22,23	computer 12:20	83:13 84:19 144:4	corresponds 45:18	<u>D</u>
44:2,3,6,6,15,16	13:9 16:16,21	144:5	47:5 114:10	d 89:4 133:7
45:2,7,19,19 46:2	26:16 27:16 61:3	context 49:21 61:2	141:10,20 142:1	data 34:1 46:19,21
47:17,18 48:3,4	66:1 67:17 68:2,4	61:4 66:2 68:2,4	142:17 143:16	46:21 47:1,4,5,7
48:17,18 49:1,3	91:18 92:2,8,9	76:12,17	155:20 157:13,17	51:10 66:4,6,7
51:7 52:9,10,21	111:14 134:17,20	continue 108:12,24	counsel 9:11 13:12	69:21 73:23 74:18
52:23 53:3,4,4,5	134:20 155:4,9	123:21 125:12	30:19 39:17 40:4	75:20,24 76:14,16
53:11 54:3,10	computing 28:16	continues 79:14	51:14 61:9 68:7	77:2 78:6,7,17,20
55:20 56:11,21	28:18 90:3 98:2	control 41:4 46:18	68:12 85:2 92:15	78:22 79:2,4,5,10
59:2,3,4,8,9,12,13	conceptual 143:23	46:24 47:16 48:2	100:17 101:2	79:24 80:1,9,10

02.11 12 14 04.5	Jacovih ad 20.10	155.01 04 156.04	division 52.12	01.20 02.2 12 12
83:11,13,14 84:5	described 20:18	155:21,24 156:24	division 53:12 56:22 158:5,11,12	91:20 92:3,12,13
84:6,18,19 85:1,3	21:16,22 23:10,22	157:14,18	, ,	92:22 93:4,15,23
85:7,8 88:12,13	24:3,8 112:9	diagram 48:8 69:17	divisions 158:1,5	94:18 95:6 96:3
125:1,1,3,8,9,10	125:23 144:3	143:23	document 8:16	97:13 98:6,13
125:24,24 126:1,2	153:1,10	dials 110:17	14:11,20 18:13,18	103:16 136:19
126:24,24 127:5,7	describes 112:6,8	dictionary 65:21	21:22 36:11,17,24	138:15
127:19 128:1,23	description 22:22	different 9:13 15:3	40:14 69:23 82:6	DX11 95:13
131:1,5,10 133:7	22:24 23:3 64:18	15:4 51:7,10 60:1	93:11 95:1,4	DX12 95:13
135:7,8,21	design 17:10,24	66:14 75:11,24	96:14,19 97:9,19	E
date 5:3 117:5,6	55:6,16 58:6	82:19 98:3 114:22	106:2 113:16,22	E 4:9 6:9 132:23
day 8:3	64:12,19 65:8,9	120:20 122:11	137:2 141:18	154:9 158:19
DC 2:9	65:14,22 67:8,9	127:7,19,24 128:1	142:3 143:5	earlier 19:19 23:24
December 9:6	109:2 127:21	130:11,12,13,21	144:20 145:3	30:23 32:8 67:24
decide 63:19	130:6,8 137:24	133:7 134:1	148:7 156:2	105:14 111:5
declaration 8:12,20	138:6,14,17	144:16 147:6	documents 96:8,12	
8:21,24 9:5,9,24	147:12 148:16	158:6,12,15	doing 28:17,21	119:21 120:14
10:3,6,23 11:3,7	151:6 155:9	differently 27:8	75:7 76:8 79:19	127:8 128:19
11:23 12:3 13:3	designed 16:1,24	diligence 107:15	81:4,9,19,22,24	136:19
13:16,22 14:1	17:1,9,9,21 18:3	direct 67:19 155:11	82:19,23,24 83:5	easily 127:21
26:10 31:9 40:8	27:14,15 30:24	157:8	84:1,1 87:4,7	education 32:17
41:2,9 44:9,23	105:15,16	directed 86:13	96:17 100:15	eight 70:21 71:3,11
46:12 47:6,10	designers 109:6	87:14	120:17,20 121:7,8	71:19,20 77:10,19
51:18 69:6 88:17	designing 17:3,6,10	direction 109:1	121:12,14,18,19	81:8,18 121:19
98:5,21 99:14	105:19 130:7	160:13	121:20 122:3,11	122:10
112:12 115:14	155:4	directly 160:20	122:18 123:6,10	either 38:6 59:2,13
117:24 145:9	designs 31:1	disagree 77:23	124:9,10 127:23	59:16 60:2 82:2
147:5 155:14	detail 119:2 126:7	114:14 132:5	127:24 129:4,21	86:24 103:23
decrease 33:9	detailed 75:12	136:15	130:4,5,6,10,11	120:9 122:2,4
34:13 35:8	details 20:17,24	disappointed 159:1	130:17,17,20,21	123:6
decreased 33:22	21:14,21,23 23:9	disclose 47:16 48:1	131:7,13,23 132:6	electrical 134:16
decreasing 35:20	23:10,21 24:1,7,8	49:15	132:6 133:6 134:5	Electronics 1:7 5:8
deduce 24:23	25:5,10,18 87:20	disclosed 72:2,6	139:1 146:21	5:23 6:1
definition 26:21	87:23 104:20	120:22 152:15	147:7	element 37:14,24
27:1,6 87:9	determine 104:15	discloses 52:8	dole 127:13	44:21 49:7 58:23
111:13 112:4,6,15		72:10 73:9,16	dominance 54:23	78:1 82:1,16
112:17 134:1	determined 80:15	111:23 120:7	dominant 55:4,15	125:7 144:4
degree 134:14,16	119:23	145:11 157:11	55:19	155:20
134:21	determining	disclosure 48:16	dot 77:17 132:22	elements 70:17
delete 9:24 15:11	115:19 116:9,22	130:15	132:22	77:3 78:13,17
demands 86:22	developed 58:10	discovery 1:18	dots 77:17	82:24 83:6,24
depending 61:19	90:20	discuss 100:7	Dr 8:21 9:20 40:4	123:5 125:2,5
depends 65:12	developing 106:8	discussed 24:22	draw 68:12	126:23 145:11,13
depicts 141:13	device 38:5 41:6,18	140:10 157:11	Drive 1:18 2:3 5:6	embodiment 82:7
deposed 6:21	41:20 42:7,15,20	discussion 120:6	due 73:5 107:15	82:11 83:3 120:7
deposition 1:13	43:3,8,23 59:1	126:4 154:13	duly 6:7 160:8	120:21 125:20,21
4:10 5:5 6:24	113:10 114:11,15	distinct 73:10	dwell 89:19	embodiments
9:15,20 104:19	114:18,19,20	distributing 130:13	DX 94:3 95:13 99:5	82:12 128:5
159:4 160:10,15	115:7,7 118:7	divided 125:5	136:20 137:10,17	employee 160:18
depositions 1:17	134:13 143:11	divisible 126:16	DX10 28:11 90:12	160:19

				Page 160
engine 112:3	ovecunts 124.19	external 31:13	122:14 127:16	25:6,19 26:4
145:14,23 146:17	excerpts 124:18 exclude 115:11	external 31:13 extrinsic 31:19	132:16,17 133:10	27:24 29:17 30:2
-	excuse 17:5 88:20	extrinsic 51:19	-	
147:3 151:17,19		$\overline{\mathbf{F}}$	133:11 141:12,13	30:16 31:3,15
152:7,10,22 153:2	excused 159:6	F 132:23	142:4,6,7,22	32:14 34:15 35:10
153:7,18,23	execute 119:9	fabricate 17:11,13	143:7,23 144:2,4	35:22 38:15 42:9
engineering 110:1	executing 119:10	107:19	144:5,18 145:5	42:17 43:5,10,24
134:16,19,20	144:8	fabricated 17:8,20	figured 48:8	45:8 46:8 48:5
engines 37:15,19	exemplary 125:20	17:21 21:5	figures 144:3	49:16 50:19 52:15
38:12 141:21	exhibit 4:10,11,12	fabricating 107:17	file 85:12	54:5,12,19 55:22
142:2,18 143:4	4:13,14,15,16,17	fabrication 17:16	filed 27:12 117:3	56:6,12 57:7,18
entire 31:12,18	4:18,19,20 8:17		filing 117:5,6	58:2,15 59:10,19
90:3,10	8:20 9:12,13,14	17:20 155:1	finally 107:12	62:1 63:14,20
entirety 13:19	9:17,20 10:17,20	face 61:13	find 110:4,19	64:4 68:18 71:24
32:11	14:12,15 18:14,17	fact 11:1 12:2	118:22 123:13	72:13 74:8,24
Environment	18:24 19:5,15,18	53:20 56:10,24	151:15	75:8,17 76:9
22:10,16	20:4,8 36:12,15	57:5 72:10 77:6	finding 110:3	78:18 81:6 83:16
Epiphany 106:10	37:1,4 40:7,15,18	102:20 128:4,7	finish 24:18 61:9	84:3 86:8 87:16
equally 125:24	41:15 69:24 70:3	135:2 151:9	61:10 62:12 81:14	88:8 89:14 90:5
127:19	93:12,15,19 96:22	factors 88:24 89:3	81:17 97:4 125:16	90:13,14 91:8
equals 149:18	113:17,20 119:16	89:4,6 99:16	finished 97:3,8	92:5 93:6 95:7,15
error 110:7,13	123:17 141:5	100:10	firm 5:16	96:5 97:15,23
155:8	142:10 148:8,10	failed 102:16	first 6:7 14:5 19:3,3	98:8 99:1,10,23
especially 134:19	148:13,14 149:6	fair 7:3,4,13,22	20:17 21:15 22:15	100:11,19 101:4
established 30:19	154:14 155:12	9:21	41:20 42:21 47:12	101:12 102:6,23
83:2	156:3	falls 51:21	47:17 48:2,17,19	103:8 104:1,16
evaluate 54:22 91:1	exhibits 13:4	familiar 98:11	48:24 49:7,21	105:2,20 106:22
91:11	103:10	101:8 103:3	50:22 52:20 65:1	107:5,8 108:2,17
everybody 134:23	expanded 152:9,13	far 153:14	65:2,3 105:18	109:11,22 110:23
evidence 31:19	expect 33:16	fashion 133:2	106:5 114:4,5	111:8,17 113:11
89:12 101:2	experience 32:17	feeds 118:13,15	118:24 121:11	114:16,24 115:9
125:14 148:11	53:16	feeling 132:4	143:10,13,16	116:4,16 120:10
exact 101:8 137:11	expert 67:4 87:13	fetch 52:10,23	144:9,9,12,16,21	121:15 129:5
exactly 43:13 45:17	90:2 91:11,17,17	119:3 157:24	149:17 156:21	130:18 132:1,13
89:18 157:20	95:20 126:18	158:5,11	first-in 64:8,9,21	135:11,15,22
examination 4:1	experts 14:10	fewer 34:20	64:23	136:9,23 137:6,20
160:7	explain 61:9 77:1	field 87:13 89:7,9	first-out 64:8,9,21	138:5,7,21 139:11
examined 6:8	82:18 112:18	109:3 134:2,18	64:23	139:22 140:6
example 19:18,23	130:1 131:2 150:6	FIFO 64:6,7,11,12	fit 34:24	141:15 142:19
20:4 41:12 46:17	explained 56:16	64:14,15,18,20	flexibility 130:8	143:18 145:19
54:16 55:11 56:10	57:22 126:7	65:5,6,11,14,16	132:7	146:14,24 147:9
57:11,11 61:17	explains 112:16	65:16,18,19,19,21	flexible 94:8	147:19 148:3,22
70:20 71:18 79:7	explanation 81:18	65:23	floating 56:2	150:7,17 151:1,11
82:10 91:14	explanations	figure 11:9 12:2,6,7	focus 32:21	152:2,18 153:3,19
107:14 110:18,19	123:20	13:7,17 29:9,12	follows 6:8	formed 87:18
119:7 135:5	explicit 48:16 68:15	29:20 41:12 69:6	foregoing 160:10	former 18:22
151:23 152:15	explicitly 47:16	69:7 70:10,11,20	forget 141:5	forming 102:21
153:10	48:1,7 49:15	71:5,16 77:8,16	form 12:15 16:19	103:6 105:1
Examples 68:17	135:3	77:20 78:11 113:4	18:1,9 19:21 21:2	formula 149:16
exams 28:9	extent 76:16	118:1 119:2	22:1 23:12 24:10	found 125:15

form detion 24.14	74.5 5 17 19 22	and 6.11 26.15		h ost 72.6
foundation 24:14	74:5,5,17,18,22	good 6:11 26:15	Н	host 73:6
25:7 54:20 68:19	75:6,7,22 77:13	35:13 39:18	H 4:9	hour 140:17
96:21	94:13 128:23,24	140:17	half 140:17	hours 14:2,3,3
four 20:12 61:19	129:4	Gouraud 150:3,5,6	hand 33:14 121:22	How's 123:21
70:24 88:24	getting 21:8 24:24	150:15 151:15,22	149:16 161:2	hundred 57:11,12
121:19,20	128:18	151:22 152:24	handed 8:19 14:14	hung 139:6
Fox 2:7 5:16	give 24:16 30:5	government 107:17	18:16 36:14 37:3	Ī
frame 116:2,13	33:23 64:18 91:13	GPGPUs 28:23	40:17 70:2 93:14	Id 149:17
117:18	123:20 124:2,17	GPU 17:24 18:3	113:19 148:13	idea 86:10 87:18
front 64:19 65:15	125:14 139:15	27:14 28:6,24	handle 58:13 88:3	106:10,11
frustrated 61:14	given 120:8 160:14	29:2,4,16,23,24	88:6,11 94:13	ideas 107:2,2,6
full 114:4 149:21	gives 109:15 133:8	35:9,18 68:6	hands 132:3	identification
157:9,10	151:14	94:17,18 103:24	happen 72:12	14:15 18:17 37:4
fully 137:17 138:2	giving 64:16	105:15,16,19	75:11 76:18	40:18 70:3 148:15
138:4,24	GL 28:8,11 98:19	106:8 137:16,18	happening 122:12	identifications
function 133:8	98:23 99:4 103:17	138:3 139:7,9	happy 27:2 106:17	
134:5 146:3,8,20	136:19,21 137:10	GPUs 28:16,20	123:21	36:15
146:22 147:7,8,16	137:11,14,18	30:4,7,24,24 35:4	hard 138:10,10,13	identified 100:21
151:18,24 152:5	138:16	35:4,7,12	138:17	100:23 118:19
152:11,14,16,21	glad 22:7 76:20	graphic 26:16 78:7	hardware 22:22,24	122:15
functional 87:6,23	77:22 139:17	78:21 83:12 84:19	23:2 86:6,17 87:3	identify 122:6
functionality 65:17	140:8	92:8,9,11 125:10	87:5,10 109:2,8	Illinois 1:15,19 2:4
65:20	go 22:7 30:11 31:7	126:13,16	111:14,24	5:7 160:5
functions 54:23	34:6 40:9 46:23	graphical 125:23	head 12:23 19:19	illogical 117:4,10
55:18 88:3 145:12	53:1 72:24 74:12	graphics 27:16,19	19:23 32:13 83:18	illustrated 41:11
145:18,22 146:12	96:14 100:6	27:20,22 28:5,19	107:7,11	image 150:2
funding 107:17,20	101:10 107:11,16	29:4 53:10 91:18	heading 20:12	impatient 128:18
further 76:1 80:23	109:2,6,7 110:10	92:2,22 94:4	heard 9:12 28:11	implement 92:12
117:24 120:3,24	110:11 115:14,15	95:20 103:22	28:12 86:2 90:12	112:16 153:14
158:17,23	117:23 124:16	118:14 126:4,5	90:17 91:20 98:19	implementation
fusion 71:11	125:15 126:6	143:1 144:8	hearsay 68:18	20:18 21:1,15,18
122:10,12 123:10	129:24 154:2	Great 32:3 34:13	93:20 94:21	21:19,20 23:21
130:3 132:4	goes 80:21 121:10	36:9 44:5	held 5:5,10	26:3 35:19 111:24
	125:20 149:18	greater 126:7	hello 159:1	112:14 148:16
G	going 7:5,12,16	ground 6:23 51:22	help 35:15 71:6	implemented 74:13
G 2:2 132:23	9:16 18:23 32:1	grounds 51:15	132:16	74:15 112:9
game 134:2	34:21 36:2,3,5	group 125:8,9	helps 88:19 110:14	128:21 153:9
gears 136:17	39:20 51:14 59:24	127:22	hereto 160:20	implied 136:5
general 28:24	60:13,21 61:10	groups 76:4 129:17	hereunto 161:1	important 114:19
29:15 34:19 35:6	68:7,11,20 76:13	guess 13:12 19:2	high 28:16,18	improve 33:12
36:1 62:8 63:1	85:15 93:18 96:18	29:8 46:4 53:9	Hilledge 6:15	improvements
67:16 68:5 86:10	96:21 101:15,15	86:16 87:12 93:2	history 103:11	33:16
88:2 92:21 105:23	105:7 120:19	97:11 107:4 118:1	Hm-hmm 11:12	include 70:16 104:8
119:8 125:9	122:24 123:1,13	120:21 128:20	20:10 24:2 125:18	145:13,22 146:12
generate 11:13	123:24 124:17,22	139:6	129:2	146:16 147:3,13
generating 13:16	132:20 140:16,20	guidelines 68:14,15	holding 69:20	150:15,23 151:17
32:24	142:23 148:9	guy 126:5	home 6:14	151:24 152:16
geometry 71:23	154:3 155:13	guys 81:15 131:11	homework 28:9	153:17
72:3,11 73:4,9,21	Goldstein 2:7 5:16		hope 109:4	included 92:15
			F	
L				

includes 46:18 55:24 80:9 93:4 118:5,11 126:8 156:13 157:1,4 including 81:20 115:7 123:11 125:10 inclusive 117:21 increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 increased 19: 120:19 intersity 150: intensity 150: intensity 150: interacting 13 Interactive 22 22:15 interested 160 interseted 12 120:19 interpretation 112:11 130: 132:9 interpretation 112:11 130: 133:21 interpretation 112:11 130: 132:9 interpretation 132:1 130:11 12:11 130: 132:1 130:11 12:11 130: 130:11 12:11 130: 130:	3:24 items 103:12 125:3	label 71:11 145:2	lifted 95:3
55:24 80:9 93:4 118:5,11 126:8 156:13 157:1,4 including 81:20 115:7 123:11 125:10 inclusive 117:21 increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 52:17 118:13,14 1: 127:6 128:3 integer 56:2 integrated 19:66:16 intensity 150: intensive 55:1 57:3 interacting 13 Interacting 13 Interacting 22:15 interface 90:1 interpreted 12:10:19 interpret 131:11 12:11 130:13:29 interpretation 112:11 130:13:29 interpreted 6:10:11:11 13:20 interpreted 12:10:19 interpreted 12:10:19 interpreted 13:10:11:13:13:13 interpretation 112:11 130:13:13:13 interpretation 112:11 130:13:13:13 interpretation 120:19 interpretation 13:29 interpretation 13:29 interpretation 13:29 interpretation 13:21 1:30:13:13:13:13 interpretation 12:11 130:13:13:13 interpretation 12:11 130:13:13:13:13 interpretation 12:11 130:13:13:13:13:13 interpretation 12:11 130:13:13:13:13:13 interpretation 12:11 130:13:13:13:13:13 interpretation 12:11 130:13:13:13:13 interpretation 12:11 130:13:13:13:13 interpretation 13:13:13:13:13:13:13:13:13:13:13:13:13:1		labeled 19:3 69:11	light 150:10
118:5,11 126:8 156:13 157:1,4 127:6 128:3 integer 56:2 115:7 123:11 125:10 66:16 intensity 150: increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 inventions 10 inventors 130 148:20 149: inventors 130 148:20 149: involved 155:19 57:6,20 147: involved 15:19 57:6,20 147: involved 15:19 57:6,20 147: involved 15:19 57:6,20 147: inv	· ·	70:11 133:12	lighting 145:12,17
156:13 157:1,4 including 81:20 115:7 123:11 125:10 66:16 interested 19: 13:21 increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 increased 19: 102:17 124:2 increased 19: 102:17 124:2 interested 19: 102:17 124:2 interested 19: 102:15 interested 19: 102:19 interpretation 112:11 130: 132:9 interpretation	T T	Labs 15:15,24 16:2	145:21 147:3,14
including 81:20 115:7 123:11 125:10 inclusive 117:21 increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9	$\frac{1}{12:13}$	lack 54:19	147:16,22 152:6
integrated 19: 66:16 inclusive 117:21 increase 33:8,19	JO 1:14 2:23 160:3	Laguna 6:15	likelihood 35:9
125:10 166:16 inclusive 117:21 increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 issue 100:3 12		language 22:23,24	limit 33:12
inclusive 117:21 increase 33:8,19 34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 intensity 150: intensive 55:1 57:3 inter 113:20 interacting 13 Interactive 22 22:15 interested 160 interface 90:1 interpreted 12 120:19 interpretation 112:11 130: 132:9 interpretation 112:11 130: 132:9 interpretation 112:11 130: 132:9 interpretation 112:11 130:	JoAnn 6:3	23:3	limitation 63:6,12
increase 33:8,19	job 61:8,15 62:20	larger 35:16,18	limited 68:16 152:9
34:14 35:8,20 increased 33:21 increasing 33:11,16 33:18 indices 91:12 interacting 13 indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 interface 90:1 indirectly 160:21 interpected 12 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 joe:22 148:10 76:22 84:8 86:19 inventing 134 100:5 130:13 inventing 134 input 65:2 inputs 111:11 112:7,10,19,23 inventions 10 inquiry 88:23 91:6 101:19 101:19 inside 69:10 77:17 77:19,21 78:10 57:6,20 147: 80:16,18 119:23 involved 155:3 120:2 122:12 involved 155:3 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 Try interpretation 112:11 130: 120:19 interpretation		LaSalle 2:13	Lindholm 40:10,21
increased 33:21 increasing 33:11,16 33:18 indices 91:12 indicia 91:5 99:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 inguiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interacting 13 interactive 22 22:15 interested 160 interacting 13 interactive 22 22:15 interested 160 interacting 13 interpretation 120:19 interpretation 122:11 130: 132:9 interpretation 112:11 130: 132:9 interpret 13: intersted 160 interacting 13 interactive 22 22:15 interested 160 interactive 22 22:15 interested 160 interpretation 112:11 130: 132:9 interpret 31: interpretation 112:11 130: 132:9 interpret 31: interpretation 112:11 130: 132:9 interpretation 12:11 130: 132:9 interpretation 112:11 130: 132:9 interpretadion 112:11 130: 132:9 interpret	127:12	launch 128:2	41:12 45:18 48:1
increasing 33:11,16	John 2:3 5:24	launching 127:20	48:16 49:12,14
33:18 Interactive 22 indices 91:12 indicia 91:5 99:21 101:3 102:21 interface 90:1 139:21 140:4,13 interjected 12 indirectly 160:21 interpet 131: industry 90:3,10 108:15 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 interpretation 45:15 46:3,5,13 interpretage 46:22 47:5,7 introduction 9 64:16 66:20,21,23 96:22 148:10 66:24 68:1,10 inventing 134 76:22 84:8 86:19 invention 89:3 100:5 130:13 invention 89:3 input 65:2 input 65:2 inventions 10 inquiry 88:23 91:6 inventors 130 101:19 inventors 130 inventors 130 148:20 149:4 inventors 130 inventors 5:6 102:17 124:3 involve 15:19 57:6,20 147: involved 155:2 instituted 51:15 instruction 44:20 Ipr 1:9 5:11 45:23 46:1 76:15 irrelevant 93: 94:3 113:13 115:8 Irvine 6:18,18 125:4 126:23<	8·15 Jonathan 2:8 5:15	law 5:16	50:3,12 51:23
indices 91:12 indicia 91:5 99:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9		lawyer 117:11	52:2 113:1,4,9
indicia 91:5 99:21 101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interpret 131: interpret 131: interpretation 112:11 130: 132:9 interpreted 63 interpretation 112:11 130: 132:9 interpretation 112:11 130: 132:9 interpretation 112:11 130: 132:9 interpretation 112:11 130: 132:9 interpreta 131: interpret 131: interpret 131: interpret 131: interpret 131: interpretation 112:11 130: 132:9 interpretation 112:11 130: 132:9 interpreta 63: interpretation 112:11 130: 132:9 interpreta 13: interpreta 13: interpret 131: interpretaion 12:11 130: 132:9 interpreta 63: interpretation 112:11 130: 132:9 interpreta 63: interpretation 112:11 130: 132:9 interpreta 63: interpretation 112:11 130: 132:9 interpreta 63: interpreta 63: interpreta 63: interpreta 63: interpreta 63: interpreta 63: interpreta 13: interpret 13: interpreta 13: interpreta 13: interpret 13: interpreta 63: interpreta 63: interpreta 63: interpreta 13: interpreta 13: interpreta 13: interpreta 13: interpreta 63: interpreta 63: interpreta 13: interpreta 23: interpreta 23: interpreta 3: interpreta 3	2:10	lawyers 14:10	141:4,10,23,24
101:3 102:21 139:21 140:4,13 indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9	20	layers 131:13	142:7 155:12
indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 120:19 interpret 131: 120:19 interpretation 112:11 130: 132:9 interpretation 12:11 130: 132:9 interpretadion 12:11 130: 132:9 interpretadion 12:11 130: 132:9 interpretadion 12:11 20: 132:9 interpretadion 12:11 130: 132:9 interpretadion 12:11 20: 132:9 interpretadion 13: 13:13: 13:13: 13:13: 13:13: 13:13: 13:13: 13:13: 13:13: 13:13: 13:13	K K	layman 30:8	157:11
indirectly 160:21 individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9)·17 Kaplan 2:12 5:20	layout 107:22	Lindholm's 51:6
individual 149:7 industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interpret 131: interpretation 132:9 interpreted 63 interpre	keep 33:15,24 36:2	leading 156:15	114:6,9
industry 90:3,10 108:15 information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interpretation 112:11 130: 132:9 interpreted 63 interpreting 133:21 introduction 9 96:22 148:10 invention 89:1 101:11,20 10 invention 89:1 102:17 124:1 112:11 130: 132:9 interpreted 63 interpreting 133:21 introduction 9 96:22 148:10 invention 89:1 101:11,20 10 invention 19 inventions 10 inventors 130 148:20 149:2 involve 15:19 57:6,20 147: involve 15:19 57:6,20 147: involves 55:21 involves 55:21 inventions 10 inventors 130 148:20 149:2 involve 15:19 57:6,20 147: involve	36:4 85:11,13	158:8	line 4:2,10 64:13
108:15 112:11 130: information 24:17 132:9 24:24 30:5 41:4 interpreted 63 44:10,14,24 45:6 interpreting 45:15 46:3,5,13 133:21 46:22 47:5,7 introduction 9 64:16 66:20,21,23 96:22 148:10 66:24 68:1,10 inventing 134 76:22 84:8 86:19 100:5 130:13 inherent 126:3 invention 89:3 input 65:2 128:6,6 135:3 inputs 111:11 149:2 101:19 inventions 10 inventors 130 148:20 149:2 inventors 130 148:20 149:2 involve 15:19 57:6,20 147:3 57:6,20 147:3 involved 155:3 instituted 51:15 56:11 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 15A 94:3 125:4 126:23 1SA 94:3 127:4 131:1,5,9 issue 100:3 12	kept 76:2 129:15	lecture 12:13,17	69:2 70:19,24
information 24:17 24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interpretio 63 interpreting 132:9 interpreted 63 interpreting 133:21 introduction 9 96:22 148:10 invention 89:3 invention 89:3 invention 89:3 inventions 10 inventors 130 148:20 149:4 involve 15:19 57:6,20 147: involve 155:21 introduction 9 96:22 148:10 invention 89:3 invention 89:3 invention 89:3 invention 89:3 invention 89:3 invention 10 inventors 130 148:20 149:4 involve 155:19 57:6,20 147: involve 155:21 introduction 9 96:22 148:10 invention 89:3 invention 9:3 invention	L occion 7.7 5.16 10	13:8,14,19 29:10	72:19 74:12 76:1
24:24 30:5 41:4 44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interpretio 63 interpreting 133:21 introduction 9 96:22 148:10 invention 89:3 102:17 124:3 inventions 10 inventors 130 148:20 149:4 involve 15:19 57:6,20 147: involve 155:2 involves 55:21 intrilevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	kind 107:24 110:21	lectures 11:15,16	78:4,16 79:16,23
44:10,14,24 45:6 45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 interpreting 133:21 introduction 9 96:22 148:10 96:22 148:10 101:11,20 10 102:17 124:2 128:6,6 135:1 128:6,6 135:1 119:23 148:20 149:2 119:25:6,6 135:1 119:21 119:25 119:21 119:5:11 119:23 119:23 120:2 122:12 120	. ₁₀ kindly 67:19	11:20,21 12:3,8,9	80:4,8,14,14
45:15 46:3,5,13 46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9	knew 99:4,5,9	30:14 32:9 69:7	83:11 119:22
46:22 47:5,7 64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 introduction 9 96:22 148:10 invention 89:7 101:11,20 10 invention 89:7 102:17 124:7 102:17 124:7 112:102:17 124:7 112:102:	115:20	leeway 68:8	120:22 121:10,10
64:16 66:20,21,23 66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 120:2 122:4 issue 100:3 12	3·19 know 6:24 7:7,18	left 16:2 71:10	122:23,24 124:17
66:24 68:1,10 76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 inventing 134 invention 89:7 101:11,20 10 102:17 124:2 128:6,6 135:1 128:6,6 135:1 128:6,6 135:1 128:6,6 135:1 128:6,6 135:1 129:2 inventions 10 inventing 134 inventing 134 inventing 134 inventing 134 invention 89:7 102:17 124:2 128:6,6 135:1 128:6,6 135:1 129:2 inventions 10 inventing 134 inventing 134 102:17 124:2 128:6,6 135:1 129:2 inventions 10 inventing 134 102:17 124:2 128:6,6 135:1 128:6,6 135:1 129:2 inventions 10 inventing 134 102:17 124:2 128:6,6 135:1 148:20 149:2 involve 15:19 57:6,20 147:1 involves 55:21 involves 55:21 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	0.0 13.17 10 12.7	left-hand 20:11	124:21 125:19
76:22 84:8 86:19 100:5 130:13 inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 120:12 84:8 86:19 101:11,20 10 102:17 124:2 128:6,6 135: 149:2 inventions 10 inventors 130 148:20 149:2 involve 15:19 57:6,20 147: involved 155:3 involves 55:21 involves 55:21 involves 55:21 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 125:4 126:23 127:4 131:1,5,9 issue 100:3 12	17.15 17 17 22.7	69:10 149:20	126:6,21,22
100:5 130:13 101:11,20 10 inherent 126:3 102:17 124:2 input 65:2 128:6,6 135:1 inputs 111:11 149:2 101:19 inventions 10 inside 69:10 77:17 involve 15:19 77:19,21 78:10 57:6,20 147:1 80:16,18 119:23 involved 155:20 120:2 122:12 involves 55:20 instituted 51:15 56:11 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 125:4 126:23 1SA 94:3 127:4 131:1,5,9 issue 100:3 12	1 22.5 10 27.7	legal 89:17,20	128:20,20 129:14
inherent 126:3 input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 128:6,6 135: 149:2 inventions 10 inventors 130 148:20 149:4 involve 15:19 57:6,20 147: involved 155:3 involves 55:21 involves 55:21 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12		100:2 101:8 103:2	135:5 149:17
input 65:2 inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9	66.1 67.17 72.11	139:24	lines 34:23 35:12
inputs 111:11 112:7,10,19,23 inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 inventions 10 inventors 130 148:20 149:2 involve 15:19 57:6,20 147: involved 155:3 involves 55:21 involves 55:21 inventions 10 148:20 149:2 involve 15:19 56:11 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	5 75:12 89:16,20	letter 132:20	119:13,18 136:3
112:7,10,19,23 inventions 10 inquiry 88:23 91:6 inventors 130 101:19 148:20 149:4 inside 69:10 77:17 involve 15:19 77:19,21 78:10 57:6,20 147:4 80:16,18 119:23 involved 155:3 120:2 122:12 involves 55:21 instituted 51:15 56:11 instruction 44:20 Ipr 1:9 5:11 45:23 46:1 76:15 irrelevant 93:1 94:3 113:13 115:8 Irvine 6:18,18 125:4 126:23 ISA 94:3 127:4 131:1,5,9 issue 100:3 12	95:17 96:2 98:6	let's 12:5 22:6	list 10:16,18,20,22
inquiry 88:23 91:6 101:19 inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 inventors 130 148:20 149:4 involve 15:19 57:6,20 147: involved 155:3 involves 55:21 involves 55:21 involves 55:21 involves 55:21 involves 6:11 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	.23 98:13,22 100:4,5	30:11 31:7 35:4	13:4 113:23
101:19 148:20 149: inside 69:10 77:17 involve 15:19 77:19,21 78:10 57:6,20 147: 80:16,18 119:23 involved 155:3 120:2 122:12 involves 55:21 instituted 51:15 56:11 Ipr 1:9 5:11 irrelevant 93: 94:3 113:13 115:8 Irvine 6:18,18 125:4 126:23 ISA 94:3 127:4 131:1,5,9 issue 100:3 12	1 100.14 100.24	61:4 74:12 76:1	listed 11:18 12:9,14
inside 69:10 77:17 77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 involve 15:19 57:6,20 147: involve 55:21 involves 55:21 involves 55:21 involves 55:21 involves 55:21 involves 6:11 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	1 110.0 11 110.17	77:24 79:20,21	13:14 19:10 31:1
77:19,21 78:10 80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 57:6,20 147: involved 155:3 involves 55:21 involves 55:21 involves 55:21 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	1 199.10 199.10	83:10 97:11 105:5	36:21 37:8 40:23
80:16,18 119:23 120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 involved 155:3 involves 55:21 56:11 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	1 124.0 11 20	107:14 110:18	70:8 89:3 149:6
120:2 122:12 instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 involves 55:21 56:11 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	135:18 136:7	113:14 115:14	listening 83:20
instituted 51:15 instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 instituted 51:15 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	137:11 140:3	122:6,9 124:22	listing 148:20
instruction 44:20 45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 Ipr 1:9 5:11 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	knowledge 11:24	125:12 126:6	lists 74:16 128:21
45:23 46:1 76:15 94:3 113:13 115:8 125:4 126:23 127:4 131:1,5,9 irrelevant 93: Irvine 6:18,18 ISA 94:3 issue 100:3 12	12:11,19 20:2,3,5	154:2 156:2	little 7:9 16:14 68:8
94:3 113:13 115:8	9 26:16,24 32:9	157:21	147:6
125:4 126:23 127:4 131:1,5,9 ISA 94:3 issue 100:3 12	48:9 117:19	level 89:17 91:10	load 94:8
127:4 131:1,5,9 issue 100:3 12	known 51:8 53:11	106:18	loaded 78:6,21
	5:21 53:16 116:10,24	levels 75:11	83:12 84:18
135:8,21 156:1 139:24	134:24	LG 1:7 5:8,23 6:1	located 5:6
157:4,12,17 issues 89:20	т	LICENSE 2:24	lodge 96:18
100000000000000000000000000000000000000	L		
<u> </u>	ı	I .	I .

	70.202.10.11.14	120.22.124.2	00 4 100 5 100 15	1,50,10,10
logic 67:14	70:2 93:10,11,14	129:23 134:3	32:4 122:7 133:15	158:19,19
long-felt 89:8	113:14,16,20	154:24	mixed 42:5	Naas 2:19 5:2
102:12	145:5 148:7,14	meant 134:12	mixing 45:15	Nader 1:13 4:3
look 12:5 19:9,15	marking 141:18	135:3,13,17,18	mode 74:19,23	5:12 6:6,13 8:21
22:6 29:8,19 32:1	142:3,5 143:5,6	mechanism 73:22	77:14 93:4 129:1	14:16 148:21
41:14 49:5 50:21	145:3	Mechell 2:13 5:19	129:3,10 134:9,12	149:7 159:4
54:23 65:21 70:10	Mary 2:19 5:2	5:19	model 20:13,17	name 5:2,11 6:11
71:5 73:18 76:1	massage 110:11,14	Media 85:16,21	21:14 23:20 93:23	132:20
76:20,23 77:7,22	massaged 109:19	141:1	94:2,18 95:6	named 149:7
77:24 78:11 79:12	massaging 155:3	memorize 73:14	97:14 107:22	152:10
79:20,21 83:10	material 31:13 32:7	memory 34:8,9	109:7,10,18 110:7	nature 53:13
86:16 94:7 97:7	52:13	38:5 41:6,18,20	110:22	nearly 90:3
104:20 118:4	materials 10:9,14	42:7,14,20 43:3,8	modeled 109:3	necessarily 51:9
119:6 123:14	10:19,22 11:1,18	43:23 51:8 59:1	modify 51:6	145:13,18,22
127:11 140:8,12	12:10,14 13:14	66:18,19 86:21,23	morning 6:11 8:1	146:16,18 153:1,6
141:23 149:20	30:11,15 31:2	113:9 114:11,15	morpho 131:2,12	necessary 62:9
156:2	36:21 37:8 40:23	114:18,19,20	131:13 134:6	need 7:17 34:24
looked 55:1 81:10	70:8 113:24	115:6,7 118:7	135:1	35:16 49:6 65:18
93:9 103:20	matter 14:6 87:19	119:2 143:10,23	morphosys 18:8	65:19 67:11 82:6
107:15 137:12,13	89:17 103:2	143:24 144:2,3,5	19:11 20:12,18,22	85:11 91:1 96:12
looking 21:4 34:7	matters 15:6 160:9	144:6,6,10,11,12	21:15 22:16 23:16	102:12 127:6
38:11 63:11 82:4	MAW 2:2	144:14,15,18,23	23:22 24:19,21	needs 73:22 89:8
90:10 101:16	Mayer 2:2 5:6,22	155:21,24 156:24	25:11,22,24 26:3	never 16:16 27:22
132:17 137:10	5:24	157:13,18	27:17,19 58:10,11	28:2,5 95:4
looks 94:4	mean 15:7 28:19	mention 52:7	107:14 109:17	new 2:9 36:3 94:3
Losoya 1:14 2:23	33:21 43:19 58:6	mentioned 28:11	131:3 148:17	106:10 110:2,6
6:3 160:3 161:8	59:21 60:5 64:15	30:23 99:13,16	Morton 36:19	nod 83:18
lot 12:19 55:3	67:20 73:12 76:12	111:6 124:8 127:1	142:10,16 143:7	nonobviousness
67:13 110:6	76:20,22 84:24	131:4 154:23	143:15	89:13 100:10
153:13 155:8	87:11 100:3 103:4	155:3	move 106:18	101:3 102:21
lunch 60:10 105:5	107:11 109:6,21	micro 126:11	moved 33:14	139:21 140:5,14
	111:2 112:23	microchips 15:19	multiple 18:7 74:13	notice 127:9
<u>M</u>	114:2 115:24	15:24	74:16 76:16,18	number 5:11 9:13
M 6:9 154:9 158:19	116:6 117:11	Microsoft 90:20	77:4 87:5 88:3	22:9 28:21 46:20
maintain 73:3	118:24 128:10	95:13	126:24 128:21	71:15 109:15
managed 144:1	130:16,24 131:7	middle 36:7	131:1,5,10,10	126:8,12 141:5
manager 119:1	131:23 135:18	midway 20:11	133:9 135:8,21	numbered 149:14
126:9 127:15,15	138:19,20 147:12	mid-to-late 115:20	142:22	NW 2:9
127:17	meaning 76:15	116:3,11 117:14	multiplication	
mapping 45:18	109:24 116:18	mind 34:17 72:20	55:21 56:1,4,8	0
150:16,24 151:5	meaningful 91:14	72:22 81:18	multi-threaded	O 6:9 154:9 158:19
151:10,19 152:1	means 7:14 17:10	132:16 157:20	29:13 51:6 127:18	oath 8:1
152:17	17:15,16,18 26:2	Minneapolis 2:14	128:2	object 12:15 16:19
mark 9:19 123:15	65:5,22 73:9,18	Minnesota 2:14	multi-threading	18:1,9,9,23 19:21
123:16 132:19	74:4,22 75:5,20	minutes 45:22	29:14,20 30:4,6,8	21:2 22:1,1 23:12
marked 8:16,20	75:23 76:7,14,18	misses 33:4,9,22	M1 20:18 23:22	23:12 24:10,10,14
14:11,14 18:13,17	77:2 81:3 82:9	34:14 35:21		25:6,6,7,19,19
36:11,14,24 37:4	83:14,23 101:15	missing 100:14	N	26:4,4 27:24 30:2
		1	L N L C . O O 1 E 1. O O	20.16 21.2 15
40:14,17 69:23	121:18 129:3,20	misunderstanding	N 6:9,9 154:9,9	30:16 31:3,15

22 14 24 15 25 10	120 11 22 140 6	01 12 02 14 02 21	124.15	07.5.7.114.0.0
32:14 34:15 35:10	139:11,22 140:6	81:12 82:14 83:21	134:15	97:5,7 114:2,3
35:22 38:15 42:9	142:19 143:18	84:5 88:10,18	opined 87:11 90:9	115:5,15 117:24
42:17 43:5,10,24	145:19 146:14,24	89:16 90:2 94:17	91:6 105:4 106:1	118:1 119:6
45:8 46:8 48:5	147:9,9,19,19	96:16,24 99:15,19	opinion 49:19	127:10,14 143:21
49:16 50:19 51:14	148:3,3,22 150:7	100:9 107:24	50:24 51:4 73:15	149:14,14,15
52:15 54:5,12,19	150:7,17,17 151:1	113:3,19 119:17	102:22 115:3	156:4,7,18,22
55:22 56:6,12	151:1,11,11 152:2	121:24 122:15	139:20 140:4,13	157:7,9
57:7,18 58:2,15	152:2,18 153:3,3	123:12,19,23	146:5,10 147:2,13	pages 136:11
58:15 59:10,19	153:19,19 156:15	124:8 125:17	147:22 155:19	paper 18:21 19:10
62:1 63:14,20	158:8	127:16 132:20,23	opinions 10:9	19:11 21:5,13
64:4 68:7,11 69:2	objections 68:15,17	133:10 137:16	103:7 105:1	22:4,6 24:3,9,22
71:24 72:13 74:8	68:21 98:15	142:9 152:13	opportunity 151:15	25:4 107:6,12,16
74:24 75:8,17	101:21 104:9	156:5,23 157:22	optimal 110:12	131:3 148:15
76:9 78:18 81:6	106:6 108:9	once 46:23 80:15	option 60:2,21,23	149:11 151:21,23
83:16 84:3 86:8	153:11 154:18	106:17 119:22	61:1 62:18	152:15 153:1,10
88:8 89:14 90:5	objective 89:4,6,12	ones 39:15	options 59:17,21	153:15,17 154:16
90:13 93:18 96:21	91:5 99:16,20	open 28:8,11 98:19	60:11 61:16,21	154:19
113:11 114:16,24	101:3 102:21	98:23 99:4 103:17	62:21 63:3 65:7	papers 12:21
115:9 116:4,16	139:2,21 140:4,13	136:19,21 137:10	104:5 111:7	paragraph 10:5,8
120:10 121:15	obtain 149:22 150:1	137:11,14,18 138:15	order 59:16 68:13	12:6 13:7 26:10
129:5 130:18	obvious 24:23		73:4 149:21 150:1 ordinary 26:14	26:13 29:9 40:9 41:2,9 44:9,22
132:13 135:15 139:11 141:15	49:19 50:15 51:5	opens 60:14 operable 37:20	•	45:13 46:12 47:6
148:9	51:24 54:1		27:5,10 49:20	
objection 43:16	obviousness 50:2	38:5,23 59:1 125:3,8	51:5 115:19 116:9 116:23 117:15	47:9,13 50:22 51:1,18,20,21
46:11 68:18,18,18	50:18 88:23 91:6	operably 125:5	126:18 134:7,10	52:5,7 53:8,9,10
68:19 69:2 87:15	91:7,10 99:18,22	operate 129:8	original 76:12,17	53:24 56:20 78:15
87:16 90:5,14,14	102:22 103:7	operates 64:24	outcome 160:21	79:18 80:8 88:17
90:21 91:8,21	102.22 103.7	operation 54:18	output 65:3 74:18	88:18 99:13
92:5,5,14,15,16	occur 75:15	55:14 56:5 72:11	77:13 128:24	111:22 112:12,18
93:6,6 94:20,21	occurred 120:2	81:3 82:2 84:2	outputs 65:2	114:1,4 115:15,17
95:7,14,15 96:5,5	occurring 80:18	124:11 126:12	outside 51:15 56:13	117:23 118:4
96:18 97:15,15,23	occurs 153:2	129:13 131:8	67:15 80:15 90:23	121:9,13 122:1,5
97:23 98:8,8 99:1	offer 94:18	operations 53:13	101:24 119:23	122:23,24 123:4
99:1,10,10,23	offered 95:13	53:21 54:4,11	139:14	124:23 136:12,14
100:11,19 101:4,4	111:20	56:11,17 57:1,3,6	overall 57:10 87:5	145:8 149:21
101:12,12 102:6	OFFICE 1:3	57:21 58:1 72:8	owner 5:7,17,20	155:13 156:21
102:23 103:8,18	Oh 7:10 14:9 16:9	72:11 75:7 81:4		157:9,10
104:1,1,16,16	67:13 107:24	81:23,24 82:17	P	parallel 73:22 74:4
105:2,20,20	okay 7:5,10,16,20	84:1 86:7,7,18,18	page 4:2,10 8:23	74:19,23 75:6,16
106:22,22 107:8,8	8:8 9:16 10:7	87:7,8 88:6,7,11	10:17 11:7,9 12:5	75:20,23,24 76:4
108:2,2,17,17,21	11:6 15:15 19:9	120:9 121:7,12,21	12:5 19:3,5,7 20:9	76:7,12,17 77:1
109:11,11,22	29:3 32:3 35:7	122:4 123:6,11	20:12 22:8 23:8	77:14 127:1
110:23,23 111:8	36:6,9 44:5 46:7	124:10 129:4	23:10,19 24:6,9	128:24 129:3,10
111:17 117:1	47:11,24 48:11	130:5 131:24	25:13 29:19,20	129:17,19,23
132:1 135:11,22	50:6,6,14,17	operative 41:20,22	31:8,10,10 32:20	130:15,17,23,24
136:9,23,23 137:6	52:13,18 53:1,9	42:21,22 49:1,2	33:3 40:9,11	131:6,7,22,24
137:6,20,20 138:5	63:9 64:10 71:17	opine 91:10 100:3	46:17 69:6,7	132:10 133:12,16
138:7,7,8,21,21	72:6 77:9 78:5,12	103:3 104:21	88:19 93:22,23	133:16,21,22
	•	•	•	•

		l		
134:3,4,9,12	percentage 57:9	Ph.D 2:8 14:17	142:1,17	139:22 140:6,16
135:6	perfect 34:3	pick 60:8	plus 44:19 45:22	141:6,15 142:19
parallelism 76:14	perform 122:20	picked 94:24	46:3	143:18 145:19
77:2 130:1	126:13 127:7	picture 11:10,13	Pluta 2:2 4:5 5:22	146:14,24 147:9
paraphrasing	145:11	piece 107:12 110:9	5:22 9:11,16,21	147:19 148:3,9,22
154:23 156:12	performance 28:16	pieces 18:3	12:15 13:17 16:19	150:7,17 151:1,11
part 48:19 49:21	28:18 33:12,19,21	pixel 38:7,13 39:1	18:1,9,23 19:21	152:2,18 153:3,11
55:5 66:13 67:6,7	54:24	39:12,15 41:5,21	21:2 22:1 23:12	153:19 154:2,10
79:16 80:23 88:3	performed 74:23	42:3,7,15,22 44:2	24:10,14 25:6,19	156:16 158:9,17
112:6 114:18,20	146:3,8,21 147:7	44:5,16 45:19	26:4 27:24 30:2	point 7:6 13:6 56:2
118:24 120:4,24	147:17	47:17 48:3,17	30:16 31:3,15	68:11 86:15
126:16 147:23	perimeter 80:18	49:1 51:9 53:4,20	32:14 34:15 35:10	106:21 123:24
151:6,18 152:7,21	period 16:14	54:3,10 55:13,20	35:22 38:15 42:9	125:17 140:17
153:22	115:19 116:9,15	56:11 59:3,9,13	42:17 43:5,10,16	pointed 142:24
partes 113:21	periodic 7:16	59:16 61:5,7,22	43:24 45:8 46:8	pointing 155:23
particular 40:8	perpendicular	63:4 74:6 79:21	48:5 49:16 50:19	portion 41:20,22
44:20 61:18 68:10	150:12	79:22 80:1,10,10	51:14 52:15 54:5	42:21,22 47:17,18
82:22 92:19 106:1	person 26:14 27:5	80:15 81:3,4,22	54:12,19 55:22	48:2,3,17,18,24
110:9 135:7	27:10 30:9 95:1	81:24 82:2 83:9	56:6,12 57:7,18	49:2 57:14,14
151:17 155:5	116:9,23 117:15	86:7,18,22 87:1,7	58:2,15 59:10,19	67:10 143:10,10
parties 160:20	126:18 134:7,10	88:7,13 94:13	62:1 63:14,20	143:13,14,16,17
passage 144:14	134:14	118:6,12,17 119:4	64:4 68:7 69:1	144:22,22
patent 1:3,4 5:7,10	personal 109:5	119:22 120:9	71:24 72:13 74:8	portions 51:8
5:17,20 10:10,11	160:12	121:7,8,12,13	74:24 75:8,17	POSITA 48:7 55:1
15:5 26:15 27:11	pertaining 1:17	122:4 123:6	76:9 78:18 81:6	134:3,15 136:7,13
29:24 36:16,19	PEs 78:6,10,21	124:10 126:3,15	81:15 83:16 84:3	position 49:22 50:1
37:5,6 40:19,21	79:19 80:22 81:4	128:8,9,12 150:2	86:8 87:15 88:8	51:21 60:23 113:8
49:4,5 52:8 53:2	83:12 84:18 120:3	158:14	89:14 90:5,13,21	113:8 114:22
58:21 70:4,4,6	120:8,16,23 121:2	pixels 33:1 44:11	91:8,21 92:5,14	124:12 128:13
86:12 111:23	121:5,6,11,14,17	44:14 54:17 80:18	93:6,18 94:20	145:17 156:20
113:10,21 141:21	121:17 122:2,3	82:14,24 83:1	95:7,14 96:5,17	157:16
143:9,17 149:1	127:21,22 128:3	84:6 120:1 122:3	97:15,23 98:8,15	possible 138:1
158:2	133:1 134:4	125:11 127:23	99:1,10,23 100:11	146:7,20,23 147:6
patents 103:11	petition 113:20	pixel-type 72:7	100:19 101:4,12	147:16
PC 69:21,22	115:6 156:3,6,18	pizza 60:9,14	101:21 102:6,23	potentially 124:9
PE 78:7,9,22 79:2,5	157:8,19	place 160:16	103:8,18 104:1,9	power 34:11 54:24
79:10,14,23 80:9	petitioner 1:8 2:6	platform 92:4	104:16 105:2,6,20	precise 95:1
80:11 83:13,14	114:14,23 115:2	please 5:14 6:11,14	106:6,22 107:8	predetermined
84:11,19,24 85:1	petitioner's 156:11	21:12 24:18 32:4	108:2,9,17,21	46:20
85:5,8 121:19	157:19	62:12 81:14	109:11,22 110:23	predominantly
127:3,5 128:7	phase 73:10,10,17	100:14 111:20	111:8,17 113:11	53:22 54:7 56:17
pen 123:15 124:1,2	74:5,7,18,22 75:6	113:15 123:18	114:16,24 115:9	57:2,23 58:7
penalty 33:5 34:4,4	77:14 128:24	141:23 157:10	116:4,16 117:1	146:5,7
34:10,11	phases 72:12 73:5	plurality 37:15,19	120:10 121:15	prefer 72:21
pending 7:20 8:11	73:11,12,20,21	38:6,7,11 41:21	124:3 129:5	prefetching 123:22
124:3	74:3 75:10,19,22	41:22 42:15,16,21	130:18 132:1,13	preparation 11:2
people 128:17	76:21,23	42:23 49:1,3 59:2	135:11,15,22	prepare 28:8
percent 55:4 57:12	phrase 86:2	59:3,13 70:16	136:9,23 137:6,20	prepared 11:16
57:13	physicists 110:4	125:6 141:20	138:5,7,21 139:11	67:20
	<u> </u>	<u> </u>	<u> </u>	

			l	l
preparing 10:23	74:6,6,17,17 75:6	proposed 135:24	99:19 105:23	26:13 41:18 47:15
11:22 13:3,24	77:3,24 78:13,17	Prosecution 103:11	106:1 116:14	58:24 70:20 88:19
presence 5:14	79:21,22 80:1,17	prove 21:5	117:6 120:21	94:2 114:5 115:18
present 2:1,18	80:22,23 82:1,16	provide 10:9 53:15	124:3,5 139:4	120:23
124:24 135:6	82:16,17,23,23,24	66:24 101:2	141:22 147:5	ready 123:23
presents 110:8	83:1,5,6,24	126:23 127:21	148:2	realism 150:2
preventing 130:6	103:22 112:3	137:19 138:3	questioning 69:3	realize 142:14
previous 29:19	118:14 119:24	139:9,20 140:13	157:23	really 49:22 87:9
31:1,2 37:24 85:9	120:3,4,8,24	provided 70:21	questions 7:6,11	87:11,19
121:21 160:6	121:1,2,12 122:2	71:3 73:15 82:10	35:24 67:22	reason 8:5,6 33:10
previously 135:24	122:3,4,6 123:1,1	100:17 124:24	153:24 154:12,19	50:21
primitive 73:4	123:5,5 125:1,2,5	125:1 137:2 140:4	154:20 155:16	recall 73:17 99:5
79:15,24 80:16,17	125:6,7,7,10,23	provides 66:4	158:17,21,23	100:13,15 102:18
80:24 119:24	126:4,8,13,14,16	97:20 111:15	quick 97:19	109:14 119:19
120:1,2,4 121:1	126:23,24 128:10	112:2,20,22		129:8 154:13,20
150:13	128:11,12,22,23	provisions 1:15	<u>R</u>	155:16 156:3,6,10
primitives 76:2	130:16 131:20,23	publicly 13:10	rastering 119:12	156:17 157:23
119:12 129:15	132:11,15 133:23	purely 125:20	rasterization 32:22	receive 37:20 38:24
prior 51:23 91:11	135:7,8 141:21	purpose 1:18 28:24	32:24 72:7,11	39:4 66:8,21
92:15 100:18,21	142:2,18 143:4	purposes 14:15	73:5,10,21 74:7	received 125:4
100:23 103:6	145:11,12,14,18	18:17 36:15 37:4	76:4,7,8 129:18	receives 66:6
117:16 140:9	145:23 146:4,9,13	40:18 148:15	129:20,21 130:4	111:15 112:1,20
probably 11:15	146:17,22 147:3,8	pursuant 1:14	130:15,17	112:22
62:14 107:19	147:18,23 151:17	put 11:4 29:9 33:23	rate 126:9	recites 143:9
117:6 135:18	152:7,10,22 153:2	44:23 65:10,15,17	read 23:7,23 24:13	recognize 18:18
Procedure 1:15	153:7,18,23	65:18 66:13	24:16,17 25:3	reconfigurable
proceed 74:19	processor 77:7,7,10	142:12	26:10 37:18 38:4	19:11 22:17 23:16
128:24	77:17 105:16,19	putting 154:18	41:3,19 47:20	24:19 25:11,22
proceeding 13:20	127:18 144:8	puzzling 25:15	50:23 51:3 67:1	148:17
proceedings 39:23	processors 77:18	p.m 140:21 141:2	67:24 72:19,20,21	reconfirmed
85:19 105:10	122:15 143:1	154:4,8 159:5,6	73:1,8 76:21	124:12
140:23 154:6	product 102:5		77:20 80:14 82:3	record 5:1,14 7:1
160:14	professional	Q	84:13,16,21,24	13:13 26:11 37:18
proceeds 77:14	139:16	qualification 35:14	96:8,11,14 97:9	39:21,24 41:3,19
process 37:20	program 35:16,18	qualified 123:9	114:9 119:21	51:3 58:24 72:21
38:13,24 39:4	69:22	qualifier 138:20	120:15,16 122:17	73:1 76:2 84:16
73:3 106:4,11	programmable	qualify 35:1,5 53:6	124:23 129:8	85:16,20 91:4
125:3,8 127:5	87:20 88:2 97:20	53:23 82:5 84:14	131:3 136:3 137:3	105:8,11 111:22
144:2 155:3,4	139:1	question 7:7,20,21	145:9 147:5 150:1	114:9 119:22
processed 80:24	programmed 28:5	12:13 21:4,7,8	157:9	120:15 122:9,14
120:5 121:1	98:3	23:19 24:7 25:17	readily 125:22	129:15 132:17
processes 77:5	programming 28:8	27:4 29:3 31:17	reading 44:12	140:21,24 145:6,9
122:21	90:19	31:23,24 34:3	74:21 75:2 78:20	154:2,3,7 157:10
processing 26:16	programs 26:17	35:1,5,6 36:8	80:13,20 83:19,21	159:5 160:13
29:4 37:15,19	project 109:17	38:17 42:18 49:10	94:16 111:22	records 122:8
38:12 53:10 70:11	projects 28:15,15	54:9 55:8 57:17	112:22 123:21	reduced 160:12
70:15,16,21,21	promise 48:15 50:2	61:11 63:23 64:17	124:13 126:17	refer 9:12,17 10:17
71:2,3,4,4,16,19	properly 25:1	67:3,5,16 96:23	134:11 157:15	46:14 47:5 123:16
71:23 72:3 74:4,5	68:17	96:24 97:12 98:13	reads 10:8 22:16	132:16 134:13
			l	l

135:20	90:15,21 91:21	represents 79:15	reviewed 10:10,15	127:10 128:14
reference 40:10	92:6,16 93:7	require 43:23 96:3	10:23 11:2 70:6	129:4,11,21
113:2 118:2 141:5	94:20 95:8,14	97:14 103:17	73:13	131:20 132:22
141:24 142:10	96:6 97:16,24	136:21 145:18	reviewing 97:2,13	133:1,3,5 135:3,4
155:12	98:9 99:2,11	required 42:2 98:7	right 6:21 12:3,10	135:9,14,21 136:6
references 32:1	101:5,13 103:18	98:14,23 99:8,17	12:14,18 13:5	137:19 146:9
117:16,20,22	104:2,17 105:21	requirement 49:4	15:16 16:5,10,11	149:16 151:22
referred 47:2	106:23 107:9	49:10,12 62:17	16:13 19:16,20	152:1 155:2
referring 9:13	108:3,18,21	73:5	20:6 21:12,16,17	156:14
10:14 18:6 25:10	109:12 110:24	requires 45:2 88:23	21:19 23:22 24:4	right-hand 22:9,13
39:1 40:10 41:10	136:24 137:7,21	90:9 104:11	28:22,24 29:4,13	69:9 149:15,17
46:16 73:20	138:8,22 139:12	139:16	29:21,22 30:15	Robert 2:2 5:22
122:16,24 129:10	147:10,20 148:4	research 32:9	31:20 33:1,6,9,17	ROBIN 2:12
133:10,11,23	148:10 150:8,18	53:17 109:1	34:4,5,8,9,11,14	Robins 5:20
136:11 137:10	151:2,12 152:3,19	researching 101:18	34:22 35:3,9	room 33:23
143:20,21 144:14	153:4,20	reservations 144:2	38:14 40:10 41:12	ROWE 2:2
144:19	relevant 13:20	reserved 144:1,10	42:8,11 43:4,9,18	rpluta@mayerbr
refers 13:8 86:5	88:22 91:6 115:18	residual 121:21	44:6,11,24 45:3	2:5
88:11 122:17	116:8	122:19	46:10 47:8,13	rules 1:16 6:23
125:9	relied 13:2	resolve 63:19	49:4,8,9,15 50:3,9	123:19
refrain 68:20	relying 52:4	resource 52:9,10,21	50:14,15 52:1,3,5	running 16:14
refresh 124:4	remainder 80:22	52:23 53:12 56:22	52:14,23 53:21	rzhu@mayerbro
regarding 156:7,17	120:2,23 121:17	98:2 143:24 144:4	54:4,18 55:21	2:5
157:23	remember 14:7	144:5 157:24,24	56:5,22 57:6 59:8	R-E-G-S 69:11
region 74:16	103:16 120:6	158:4,5,11,11	59:18 60:3,12,16	K-E-G- 5 07.11
128:22	137:5 153:15	resources 29:1,2	60:19,20 61:24	S
regions 76:3 129:16	reminds 68:13	46:19	62:23 63:7,16	S 4:9
register 66:17,19	render 51:24	respect 14:6 27:6	67:14 71:18,19	satisfactory 124:19
67:2,6,9,11,24	Rendering 148:16	124:19 128:4	72:12 73:10,13,16	satisfied 27:7
69:14,16 84:7	repeat 25:9	145:10 156:21	74:7,23 75:7,16	satisfies 44:15,20
85:12	repeated 42:11	158:1	76:8 77:7,11	45:7 49:12
registers 85:12,13	49:7	respectfully 55:9	78:17 79:10 81:5	satisfy 27:1 139:7
related 10:10 29:14	repeating 21:9	respective 80:10	81:24 82:2,11,17	saw 131:4
34:2 36:1 51:20	48:20	125:3,8	83:2,7,15 84:9,11	saying 39:11 43:12
51:22 95:17 97:18	report 27:2 46:14	Respondent 1:11	85:6 86:7,13,18	44:14,17,18,23
105:24 125:10	46:17 49:18 56:13	2:11,16	86:22 87:1,12	45:6,12,14 46:1
relates 135:6	67:18 68:9,10	response 123:9,9	89:4,21 91:7 92:4	50:2,8,9 51:22
relating 26:14	89:17,21 90:9,24	responsible 127:18	92:13 93:5 94:19	65:4 78:16 79:6
29:12	91:4 95:18 97:18	127:19 152:5,22	95:20 99:20	85:7 110:14
relation 144:3	100:4,9 101:24	rest 79:12	100:10,18 101:1	118:10 121:4
154:14	102:8,13,15,19	restate 69:1	102:22 103:24	122:9,10,11 123:4
relative 160:17,18	116:1 140:8,12	result 115:18	104:8,21 105:4,24	141:20
relevance 13:19	reported 2:23	results 106:18	108:5 110:10	says 15:1,2 20:16
18:10,24 22:2	160:11	resume 16:15 28:2	111:7,21 112:13	21:13 22:6 23:1
23:13 24:11,22	reporter 6:2 160:1	136:18	112:15 114:3	23:16 24:18 25:22
25:7,20 26:5	160:4	retain 46:19	115:8 116:15,19	26:7 29:22,23
58:16 67:18,23	represent 107:13	retire 46:23	117:6 118:2 120:9	32:21 38:23 39:3
68:11,19 69:2	representing 5:2	review 97:19	121:3,7,14 122:13	41:3 42:20,24
87:15 88:20 90:6	6:3	113:21	123:2 124:11	43:18 44:2 45:4
07.13 00.20 70.0	0.5	110.21	123.2 12 1.11	
	<u> </u>	<u> </u>	<u> </u>	<u> </u>

46:18 49:18 50:22	20:20,21 22:11,19	September 117:3	134:21,24 135:1,3	130:21 131:11
50:23 52:8 53:6	25:15 26:19 32:1	set 65:20 86:17	135:3,7,13,14,17	132:6,23,24 133:3
53:16 56:23 59:14	37:16,22 38:2,9	87:3 94:3 144:7,9	135:17,21 136:1,4	133:4,7 134:6
62:17 69:9 71:11	39:7 41:7,8,24	144:12 161:1	136:8,12,13,13	142:21,23
71:21 73:11,20	47:19 51:1,11	shader 86:3,5,13,14	148:17	slices 81:9,9,19
74:3,13 75:2,3,15	52:11 59:5,6 62:4	87:10,14,20,22	SIMDs 133:9	120:20 131:10,17
75:22 77:13 78:6	67:22 69:12 70:13	88:2 90:4 93:4,23	similar 76:3 129:17	131:19 142:22
78:20 79:11,20,21	70:23 71:10,13	94:2,12,12,18,19	simple 61:20	slow 7:8
80:9,13 83:22	73:7 74:1,20 76:5	95:6 96:4 97:14	simulate 106:12,20	slower 34:9
85:5 88:10,12	77:16 78:8 79:12	97:14,20,22 98:7	107:16 109:7	SM 94:2
89:10,18 93:23	80:2,6,12,19 81:1	98:14,24 99:8	110:7	small 34:24 57:9,14
94:8,11 100:6	87:20 88:14 89:1	103:17,24 136:22	simulated 107:24	smaller 34:20
103:4,4 110:9	89:22 93:16,24	137:19 138:1,4,16	153:13	software 22:10,16
112:20 115:24	94:5,9,15 97:11	138:20,24 139:9	simulating 106:11	107:13 111:14,24
116:12 118:5	101:10,19 102:4	139:10	106:20	solution 110:12
119:1,7 121:6,8	103:11 114:6,12	shaders 98:11 99:5	simulation 153:18	139:1
121:11,13 122:1,5	115:21 118:8,21	99:6	155:5	somebody 126:17
122:18,23 123:8	118:22 124:16,18	shading 88:3 98:3	simulations 107:20	134:2,18,19
124:20 126:19	129:19 133:14,19	145:12,22 146:2,8	107:21 108:7	147:12
127:11 128:20	137:13 145:15	146:12,20,21	simulator 106:13	sorry 27:7 40:9
129:15 131:15,22	148:18,21 149:18	147:3,7,8,14,23	106:15	55:10 68:2 70:24
132:10 133:6	149:23	150:3,5,6,15	simultaneously	83:19 89:23 90:13
135:6,24 137:3	seen 14:20 16:15	151:9,15,22,22,24	122:21 127:22	92:14 100:3 106:6
146:16,18 149:17	36:17 95:4 113:22	152:6,14,16 153:1	128:8,10	122:17 126:22
150:10	113:23,24	shaked 61:13	single 31:13 68:16	133:14 140:1
scheduling 68:13	select 38:5 59:1,7	shape 79:16	76:15 85:1 86:17	142:14 143:22
school 16:8	59:12,16,24 60:1	shelf 130:3	87:3 124:10	149:4,4
science 134:17,20	60:2,10 61:15	short 105:9 140:22	126:23 131:1,4,9	sort 15:19 55:24
scope 51:15 56:13	62:21 64:2,3,6,11	154:5	135:8,20	67:14 71:10
67:15 68:9 90:24	64:12,14 65:6	Shorthand 160:3	sir 6:12,20 8:19,22	106:12
139:14	67:14 69:19 88:13	show 8:14 72:15	13:23 15:17 16:3	sorted 76:3 129:16
search 46:15	selecting 60:24	83:8	18:16 19:7 20:23	South 1:18 2:3 5:6
second 41:22 42:22	61:6 111:11	showed 82:4 83:9	24:13 30:13 31:5	so-called 51:23
47:18 48:3,14,18	112:19	85:12 104:22	36:5 37:3,12	space 144:2,10,11
49:2 52:22 58:23	selection 60:16,17	showing 69:16	58:22 61:12 70:7	144:13,14,15,18
94:7 115:17	60:20	shown 30:15 70:11	71:8 85:24 105:14	spaces 143:24
121:13 143:10,14	selects 86:24 111:6	70:20 71:15	124:8	144:6
143:17 144:5,7,15	112:7,10,23	118:21 119:1	size 33:8,11,16,19	spans 31:9
144:15,22,23	sentence 20:16	127:20 132:21	34:14 35:8,13,20	speak 7:1,2
157:8,10	21:13 22:15 23:7	133:12 144:4	76:4 110:19	speaking 68:21
secondary 91:12	23:8 24:13,16,18	shows 77:16 142:22	129:17	speaks 28:2 145:21
100:10	48:20 49:21 73:19	143:23	skill 26:14 27:5,10	specialized 87:10
section 31:12,18,20	79:11 80:8 94:11	side 22:9,13 149:15	49:20 51:5 115:20	specialty 91:13
32:12 51:22 83:8	94:16,24,24 95:3	signature 8:24 9:3	116:10,23 117:15	specific 29:18
128:15 144:6,20	95:3 100:6 114:5	signed 9:5	126:18 134:7,10	127:5
sections 57:4	115:17 118:5	SIMD 127:2,23	slice 81:10,14,22,23	specifically 28:14
see 8:6 10:12,17	120:23 121:6,11	129:7,8,12 131:1	82:19,22 123:3	154:14
11:11 13:18 14:18	121:13	131:2,2,3,4,4,9	128:3 129:12,22	specified 22:17
16:9 19:13 20:14	separate 51:7	133:2,5 134:5,13	129:23 130:9,20	23:17 24:20 25:23
	_			
	•	•	•	•

160.16	52-22 110-2	20.10 (0.17 (2.22	25.2.27.6.46.6.12	156.17.150.00
160:16	52:22 119:3	39:18 60:17 63:22	35:2 37:6 46:6,13	156:17 158:22
spend 13:24 104:19	stretch 133:24	72:18,23 73:19	48:13 61:2 62:23	160:14
stack 131:11	strike 28:10 42:4	78:3 82:6 84:17	64:15 72:15 77:14	text 118:23 124:13
stand 23:2,5 51:13	48:22 55:12 71:22	93:3 96:9 105:6	78:9 81:15,23	124:15 149:14
stands 22:22 69:14	74:14 131:18	111:21 113:7	86:17 121:10	textbooks 12:20
start 42:4 74:14	143:14	122:22 126:2	131:16,17,19	texting 151:10
88:21 110:12	structure 126:24	137:23 140:18	132:5,10 133:15	texture 52:10,14,23
134:9 141:22	Studdard 134:8	surface 150:12	133:16,22	53:3 56:4 57:1,6
started 16:9	students 18:22	swear 6:3	talks 33:4 42:12	57:14,20 58:1
starting 40:11	108:20,24	switch 36:3 157:21	52:13,17 53:4	145:13,18,23
119:22 122:23	studied 92:18	switching 58:11	91:5 92:13 128:5	146:3,8,13,17,22
128:20	stuff 19:19 32:12	111:5 136:17	134:8 135:24	147:2,8,17,23
starts 31:8 121:10	33:23 110:21	sworn 6:8 160:8	136:12 142:21	150:16,23 151:5
149:21	153:14	system 20:13,17	target 29:17	151:16,16,19,24
start-up 16:15	Stuttard 70:4,10	21:14 22:17 23:16	taught 28:7	152:6,10,16,21
state 6:11,14 13:13	71:21,23 72:3,7	23:20,20 24:19	TCB 44:19,23 45:6	153:2,6,18,23
46:19,21,23 47:1	72:10 73:9,13,16	25:11,22 51:6	45:22 46:1,18	157:24 158:5,11
47:4,4,7 85:11,13	75:13,15 77:6	58:13 72:2,6 78:7	teach 108:20,24	thank 20:6 71:7
120:18 122:20	78:16 79:6,9,10	78:21 83:12 84:19	teaching 53:17	73:2 86:1 124:6
160:4	79:20 118:2,11	88:2 109:19 119:8	technical 31:7,9	133:10 141:7
stated 68:17	119:4,15 131:17	systems 26:17	32:11	145:4
statement 16:7	131:20 133:22	54:23 73:6	techniques 125:22	Thanks 80:5
23:15 24:24 31:19	134:11,12 135:2		Technologies 1:10	thereof 1:16 111:15
44:12 47:21,22	135:13 145:10		5:9,17,21	112:1
51:13 53:18 77:23	Stuttard's 130:14	T 4:9 6:9 154:9	tell 14:4 25:13	thing 42:2,6 45:24
84:8 88:14 115:23	style 109:5,6	158:19	29:22 64:20 74:10	65:1,2,3 82:3
156:11	sub 60:10	take 7:16,21 36:4	75:10 80:3 87:21	84:23 108:13
STATES 1:3	submitted 8:12	39:17,19 49:20	90:8 91:1 94:23	115:12 120:17
stenographically	98:5,21	57:10 76:24 80:23	96:8 100:2,4	130:20 137:14
160:11	subordinate 50:23	85:14 106:11	109:10 110:21	things 11:22 34:20
step 105:18 106:5	subset 46:20	109:15 120:3,24	117:21 124:14,15	43:3 59:8 60:1,3
121:22	substance 40:5	136:14 140:12,18 155:13	137:23 140:12	66:9 67:13,21
Sterne 2:7 5:16,18	158:22		telling 17:3,6	75:23 76:18 79:3
stop 59:4	subway 60:13,14	taken 1:14 5:7	temporary 69:20	79:4,8,13,19
stopped 97:9	60:22 61:3	39:23 85:19	ten 130:3	81:10,19 82:19
storage 46:19	success 89:8	105:10 140:23	term 68:16 92:21	99:17,21 109:19
store 41:21,22	sufficient 21:1	154:6 160:15	101:8	110:2,5,6 120:20
42:21,23 44:2	35:14 96:13	talk 25:9 35:4 40:4	terms 54:24 55:3	122:11 123:11
49:1,2 51:6 66:20	117:18	52:5 61:4 67:20	55:16 57:9	124:18 128:1
66:22 126:11	Suite 2:14	67:21 83:4 119:20	terrible 141:22	130:11,12,21
stored 42:3,6 43:3	supervised 28:15	130:8 150:21	testified 6:8 13:18	139:8
43:8,22 44:10	supervising 28:15	151:4,21,23	105:14 155:7	think 10:1,16 12:13
45:6,24 80:10	supplied 127:4	158:21	157:3	13:5 23:4 29:15
stores 41:4 42:15	support 26:18	talked 23:8 106:18	testify 160:8	29:16,16 30:19
43:23 44:24 47:1	31:19	119:18 125:21	testifying 7:24	31:24 32:8 45:14
48:17,18 52:9,10	supposedly 95:22	127:8,8 128:19	testimonial 68:14	45:21 47:20 49:18
65:1 78:17 86:21	95:24	136:19	68:15	50:4,7,14 53:18
storing 47:17,18	Supreme 1:16	talking 23:11 24:21	testimony 40:5	54:9 56:15 65:5
48:3,4,24 52:21	sure 7:8,10,10,23	25:12,14 30:6,8	152:24 154:24	72:18 76:11,16

77:2 103:13 109:4	tie 53:7 68:9	110:13 155:8	95:5,11,19 96:10	131:14,14 144:10	
110:17 111:5,19	ties 14:9	triangle 79:17	97:1,21 98:4,12	144:13,15,16,23	
· ·		80:16 119:23			
111:20 117:10 129:7 131:10	time 5:3,13 7:17		98:18 99:7,12	types 43:7 51:7,10 59:8 125:24 127:7	
	13:24 18:24 21:24	150:11	100:8,16,22 101:9		
134:5,15,22	27:11 34:10 39:18	trick 43:2	101:17 102:2,10	127:19 128:1	
141:13 142:8	39:21 40:1 57:12	tricks 61:20	103:5,15,21 104:6	133:7 158:6,12,15	
153:13	59:18 60:11 75:11	tried 28:16 82:18	104:13,23 105:5	typewriting 160:12	
thinking 109:5 third 94:11 149:13	76:19,24 82:20,20	true 47:21,23 53:18	105:13 106:3,7	typical 35:18	
	83:20 85:16,21	68:6 99:20 109:3	107:3,23 108:6,14	134:22	
149:14	99:9 104:20 105:8	134:22 146:2	108:19 109:9,20	U	
thorough 104:11	105:12 115:19	155:7 160:13	111:4,12 112:5	ULC 1:10 5:9,17,21	
thought 9:11 62:12	116:2,8,13,14,18	truth 160:8	113:14,18 114:21	ultimately 107:21	
125:16 128:17	116:18,19 117:18	try 7:2,5 43:7 71:5	115:4,13 116:7,20	understand 6:20	
thread 37:21 38:6	133:9 137:9	101:10	117:8 120:13	7:6,24 21:9 39:8	
38:13,14,24 39:5	140:21 141:1	trying 6:24 30:7	121:23 124:5,7	39:12 45:16,17	
39:8,13,13,16	152:9 154:1,4,8	42:14 43:2 45:16	129:9 130:22	59:12 43:10,17	
41:4,10 44:15,16	158:18,23 159:5	45:17 50:18 53:7	132:8,18 135:12	78:23 88:20,22	
46:18,19,24 47:15	160:16	62:13 81:17 82:21	135:19 136:2,16	91:14 107:5 113:7	
48:1,2 59:2,13	times 6:21 71:11	110:5	137:4,15 138:6,12	122:9 133:19	
111:16 112:2,3,21	77:19 116:22	Tuminaro 2:8 4:4,6	139:5,19 140:2,11	134:3	
113:5,8 114:6,10	135:2	5:15,15 6:10 8:18	140:18 141:3,7,8		
114:15 119:1,4,7	timing 34:4	9:14,18,22 13:1	141:16 143:2	understanding	
119:8 126:9 127:9	tired 36:6	13:12,21 14:13	144:17 146:1,19	88:1,4 89:18	
127:11,12,15,15	title 14:15 19:10,15	16:20 18:5,15	147:4,15 148:1,12	understood 7:12,15	
127:17 130:12	titled 8:20 10:19	19:1 20:1 21:6	149:3 150:14,22	31:24	
144:9,15,16,23	148:15	22:5 23:18 25:2	151:8,20 152:12	unfortunately	
155:16,19 156:7	today 8:3,8 119:21	25:16 26:1,8 28:4	152:23 153:8,16	89:23	
156:13 157:1,12	137:16,16	30:10,20 31:6,16	153:24 156:15	unified 86:2,5,13	
157:16	Today's 5:3	32:19 34:18 35:17	158:8,20 159:2	86:14 87:9,14,22	
threads 38:7,8 39:1	told 50:17	36:2,9,13 37:2	turn 8:23 10:5 11:6	88:1 90:4 93:4	
39:2 41:5,21,23	top 11:9 31:10 69:7	38:18 39:18 40:2	19:2,4 20:8 22:8	94:4,12,19 96:3	
42:3,4,7,8,15,16	71:10 93:23 118:1	40:16 42:13 43:1	26:9 32:20 33:3	97:14,22 98:7,11	
42:22,23 44:3,3,6	119:7 124:23	43:6,14,21 44:4	37:10 40:3,7 47:9	98:14,23 99:4,6,8	
44:6,7,11 45:2,7	127:10,16 133:11	45:11 46:9 48:10	51:17 58:20 69:5	103:17,24 136:21	
45:19,20 46:2,20	topic 134:17 154:20	49:23 50:20 51:16	70:19 72:18 78:1	137:19 138:1,4,16	
47:17,18 48:3,4	157:21	52:19 54:8,15	88:16 93:22 113:1	138:19,24 139:8	
48:17,18 49:2,3	topics 36:1 58:11	55:7 56:3,9,19	114:1 119:15	139:10	
51:7,9,9 52:9,10	111:5	57:15,19 58:9,19	141:4 142:9 145:8	unit 77:17 118:5,11	
52:14,14,21,22,23	Tough 67:3	59:15,23 62:5	149:13 157:7	119:3 126:10	
53:3,4,5,5,11 54:3	track 33:24 46:22	63:17 64:1,5	turning 23:7	UNITED 1:3	
54:10 56:21 59:3	85:11,13	68:12 69:4 70:1	two 35:12 43:3,7,12	units 87:6,23 126:9	
59:4,8,9,14 61:22	TRADEMARK 1:3	72:1,17 74:11	43:12,13,15,18	126:12	
61:22 63:4,4	tradeoffs 110:1	75:4,14,21 79:1	59:8 60:1,3 61:18	university 6:18	
118:6,12,18,18	transcript 160:11	81:11,21 83:17	62:18 131:13	107:18	
119:10 144:7,7,8	transferring 73:23	84:10 85:14,23	144:6	unusual 107:18	
144:9,12 158:4,6	transfers 79:24	86:11 88:5,15	two-dimensional	unwise 109:1	
158:10,12,14,15	transitioned	90:1,11,18 91:3	79:16 119:9	use 15:5 17:19,19	
158:15	137:12	91:16 92:1,10	type 35:2,18 54:18	28:16 35:19 60:7	
three 34:23 61:19	trial 1:4 5:10 110:6	93:1,10,13,21	55:13 56:5 131:8	86:14 107:14	

110:18 130:2 26:2 106:19 154:11 108:4,10,22 90:8 97:6 103:20 144:7 107:21 108:8 109:13,23 111:1,9 108:5,23 110:17 went 16:8 24:6 uses 90:3 154:14 103:13 107:20,21 111:18 113:12 111:2 114:8 121:8 usually 56:2 video 126:1,2 108:8 126:5 114:17 115:1,10 133:14 136:1 **U.S** 10:10 36:16 Videographer 2:19 we'll 96:14 116:5,17 117:2 139:2 143:8 145:7 37:5 40:19 70:4 5:1 6:2 39:20.24 we're 6:24 28:17 120:11 121:16 151:4 113:21 85:15,20 105:7,11 29:1 39:20,24 129:6 130:19 year 137:11 140:20,24 154:3,7 50:8 58:11 85:15 132:2,14 135:16 years 12:21,22 16:4 \mathbf{V} 159:3 85:20 105:7,11 135:23 136:10 32:10,16 V 23:5 videotaped 159:4 110:2,5 121:9 137:1,8,22 138:9 **Yep** 33:2 51:19 verilog 22:24 23:1 view 50:8 52:2 155:15 157:20 131:16 132:17 138:23 139:13,23 23:2 views 10:9 133:15,16 140:20 140:7 141:18 York 2:9 **Veritext** 5:2 6:3 visiting 101:15 140:24 143:6 142:3,20 143:5,19 Z version 96:3 98:23 **Vitae** 14:16 148:24 154:7 145:3,20 146:15 zero 119:7 136:20,20 137:17 vs 1:9 159:5 147:1,11,21 148:5 137:18 152:9,14 zhe@skgf.com We've 140:16 148:23 150:9,19 \mathbf{W} versions 95:12 2:11 **WHEREOF** 161:1 151:3,13 152:4,20 versus 5:9 Wacker 1:18 2:3 **Zhu** 2:3,8 5:18,24 wish 46:14 125:16 153:5,12,21 159:6 vertex 38:7,13 39:1 5:6 5:24 Withdraw 90:14 160:7,7 161:1 39:13 41:5,23 want 13:13 15:9,11 witness 4:2 5:11 word 17:15,17,20 0 32:17 35:15 36:4 42:8,16,23 44:3,6 6:4,7 12:16 18:2 47:12 60:7 68:16 **00742053** 113:21 44:11,15 45:19 36:4 39:17 44:13 18:11 19:22 21:3 86:14 110:18 **053** 10:11 26:15 45:5 49:24 50:11 47:18 48:4.18 22:3 23:14 24:12 work 6:17 15:18,19 27:11 29:24 30:4 49:3 51:9 53:5,11 55:17 57:10 58:7 24:15 25:8,21 62:10 67:9 92:3 37:5 49:4,5 51:24 53:13,22 54:7 60:9 64:17,18 26:6 28:1 30:3,18 117:12 133:2.7 52:8 53:2 58:20 55:2,18 56:17,20 65:12,13,22,23 31:4 32:15 34:16 139:17,17 86:12 111:23 56:24 57:5,20,24 67:17 73:18 76:24 35:11,23 36:6 **worked** 15:15 113:10 141:21 59:3,8,17 60:5 77:22 82:5,8 83:4 27:22 139:15 38:16 42:10,19 143:9.17 158:1 61:7,22 63:4 78:8 92:3.11 97:4.9 43:11,17 44:1 working 26:15 77:3 **084-002437** 2:24 78:17,22 79:2,4,5 103:3 104:7 113:7 134:2,4,18 45:9 48:6 49:17 79:10,15 81:20 121:24 122:8,22 52:16 54:6,13,21 **wrapping** 139:24 1 82:17 83:13,14 123:16,24 132:4 55:23 56:7,14 write 13:22 25:1 **1** 4:11,15 8:17,20 84:1,2,6,9,20 85:1 136:14 57:8 58:4,17 35:12 67:1 68:1 9:12,14,20 10:17 85:3,7 86:7,18,21 wanted 9:19 31:23 59:11,20 62:2 107:12 141:17 40:7 41:17,18 131:12 135:20 86:24 87:7 88:6 63:15,21 68:8 143:3 42:2,6 43:8 44:21 wants 66:22 94:17 88:11,12,13 94:13 72:14 74:9 75:1,9 **writing** 106:19 49:4,7,8 50:3,5,12 118:6,11,18 119:4 103:23 written 12:21 18:21 75:18 76:10 78:19 53:1,2 58:21 122:21 123:11 Washington 2:9 81:7,17 84:4 86:9 62:15 76:2 129:16 61:21,24 63:7 124:9 126:3,15 wasn't 27:4 87:17 88:9 89:15 wrote 26:22 86:16,20 87:13 128:11 130:5 wav 15:7 58:7 90:7,16,22 91:9 88:6,10 113:10 X 150:12 158:14 61:13 76:21 94:4 91:22 92:7,17 135:5 142:22 vertices 44:14 96:2 98:6,22 **X** 2:3 4:9 6:9 154:9 93:8 94:22 95:9 143:7,9 144:3 82:15 83:4,5,6,9 100:7 127:20 158:19 95:16 96:7,23 **1:49** 140:21 83:10 119:11 130:1,7 131:2,12 97:17 98:1,10,16 Y **10** 4:4,5,20 11:7,9 125:11 127:20 138:18 140:3 99:3 100:1,12,20 24:6,9 29:20 33:3 yeah 11:17,20 15:4 146:10 128:8,10 101:6,14,22 102:7 69:6,7 70:21 71:2 19:6 26:12 28:9 VHDL 22:18,21 ways 139:1 103:1,9,19 104:3 71:2,4,4 77:7,15 31:23 33:20 37:13 Welcome 40:3 23:1,2,10,17 24:8 104:10,18 105:3 79:23 80:4 94:4 39:6 55:2,4 59:14 24:20 25:5,18,23 85:24 105:14 105:22 107:1,10 119:18 120:22 62:8,22 67:15

101 0 100 00	07.5.7.0	2 4 12 10 14 17		
121:9 122:23	97:5,7,9	3 4:13 18:14,17	5 4:15 37:1,4,11,14	9
126:8 148:8,13,14	140 142:24	19:15,18 20:4,8	38:23 41:15 88:19	9 4:19 9:6 78:2,15
149:6	148 4:20	23:8,10,20 70:10	119:13 143:23	83:11 113:17,20
10-10 22:8,12	15 4:16 143:20	70:11 71:16 77:16	144:2,4,18 145:5	156:3
10-3 19:3 20:9	161:2	78:11 122:15	50 14:3	9:05 1:19
10:09 40:1	154 4:5	127:16 132:17	505 144:7,13,14,18	9:07 5:4
100 51:18,20 52:5,7	158 4:6	133:10,11 141:1	51 111:22 112:15	9:55 39:21
1003 9:17	17 4:11,19 128:20	154:14	125:19	90 57:13
101 53:10,24 56:20	170 142:24	3D 28:5,19 29:4	52 112:12,16,18	93 4:18
102 119:1 126:9	18 4:13 72:18 74:12	119:10	125:19	
127:15	77:13 128:16,19	3:06 154:8	520 144:6	
1024 118:5,11	129:14 133:23	3:12 159:5,6	55402 2:14	
119:3	134:9,12 136:3,8	30 16:4 79:23 80:4	56 119:13	
104 52:8,21 126:9	19 19:5,7 88:17,18	129:14	57 78:4,16 83:11	
105 126:11	99:13 128:20	31 12:6 13:7 29:9	58 126:22	
106 52:9,22 70:12	1984 16:2	312 2:4		
70:22 71:3,11,15	1987 16:9,12	33 70:19 71:1,2	6	
74:4,6,17 75:7		136:11	6 4:4,16 40:15,18	
77:15,18 78:11		349-0172 2:15	141:6 155:12	
79:5 81:24 82:1	2 4:12 10:5,8 14:12	36 4:14	60606-4637 2:4	
83:24 84:1 122:15	14:15 19:5 25:13	37 4:15	612 2:15	
122:18 123:1,3	32:21 51:22 70:20	371-2600 2:10	64 119:14	
124:9 128:22	85:16,21 119:10	38 26:10,13	67 143:20	
129:4,11,20	124:17 125:19	39 80:9	69 4:17	
130:16 131:20,23	144:3,5		7	
132:11,15 133:11	2B 71:5 77:8,9	4	74:17 22:9 31:8,10	
133:23 134:5	2D 119:9	44:14 36:12,15	41:17 69:24 70:3	
106s 71:20	2:12 141:2	41:12 70:19 113:4	115:15 119:10,16	
1061 70:17 78:13	2:32 154:4	118:1 119:2 126:6	145:10	
126:22 127:3	20 4:6 40:11	142:6,7,10	70 55:4	
145:11,13	200 143:24 144:4	4.0 93:23 94:2,2,18	701-8641 2:4	
1069 126:10	20005 2:9	95:6 97:14	701-8041 2.4 7015913 40:19	
108 126:10	2003 115:20 116:3	40 4:16 14:2,3	7013913 40.19 71 1:18 2:3 5:6	
11 12:5 28:11 72:19	116:11 117:3,14	117:24 118:1		
138:15	134:11,24	410 156:1 157:12	72 40:9,9 41:2,9 44:9,22 45:13	
11:15 85:17	2014 9:6	415 6:15	46:12 47:6 155:13	
11:36 85:22	2015 1:20 161:2	42 47:16 80:14	7233335 36:16	
1100 2:9	2015-00325 1:9	119:18	73 47:9 50:22 51:1	
113 4:19	5:11	420 41:4,11 46:18	73 47.9 30.22 31.1 7363427 70:4	
116 117:23 118:4	202 2:10	47:1 113:5,9	7742053 10:11 37:5	
12 4:12,14,18 31:10	205 144:10,11	114:10 155:17	// 444033 10.11 3/.3	
12:06 105:8	21 135:5	156:8,14 157:12	8	
12:50 105:12	22 115:15,16 157:7	43 119:6,22 121:10	8 4:11,18,20 32:20	
125 144:8	24 4:17	122:23	73:19 93:12,15	
13 114:2,3 115:5	25 32:16 46:17	45 80:14 120:22	8-14-2015 5:3	
156:4,7,18,22	126:6	47 119:19 121:11	80 55:4 57:12	
131 145:8	2800 2:14	122:24	800 2:13	
14 1:20 4:12,13	29 76:1	48 143:20	84-002437 161:9	
93:22,23 96:23	3	5	87 10:17	
			3. 10.11	

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

FOR THE NORTHERN DISTRICT OF CALIFORNIA

IN THE UNITED STATES DISTRICT COURT

UNIRAM TECHNOLOGY, INC,

No C 04-1268 VRW

Plaintiff,

FINDINGS OF FACT AND CONCLUSIONS OF LAW

TAIWAN SEMICONDUCTOR MANUFACTURING COMPANY, LTD and TSMC NORTH AMERICA,

Defendants.

This is a patent dispute between UniRAM Technology, Inc ("UniRAM") and Taiwan Semiconductor Manufacturing Company Ltd and TSMC North America (collectively "TSMC"). UniRAM sued TSMC for patent infringement and misappropriation of trade secrets, among other claims. TSMC counterclaimed that one of UniRAM's patents was unenforceable due to inequitable conduct in front of the Patent and Trademark Office.

On January 14 and 15, 2008, the court heard testimony concerning TSMC's allegation that UniRAM committed inequitable conduct in the prosecution of US Patent No 6,108,229 ("the '229 patent"). The court heard live testimony from Bo-In Lin, Jeng-Jye

> ATI 2075 LG v. ATI IPR2015-00326

Shau, David Taylor, and Carl Sechen, and deposition testimony from Richard Killworth and Peter Gillingham. That testimony supplemented testimony heard by the court from September 9 to 21, 2007, at the jury trial of UniRAM's trade secret misappropriation (and related) claims.

5109).

FINDINGS OF FACT

- 1. United States Patent 6,108,229 ("the '229 Patent," Ex. 353) issued on August 22, 2000, from United States Application Serial No 09/114,538 ("the '538 Application," Ex. 5133), which was filed on July 13, 1998. The '538 Application was filed as a continuation-in-part ("CIP") both of United States Application Serial No 08/805,290 (the "'290 Application," Ex 5132) and United States Application Serial No 08/653,620 (the "'620 Application," Ex
- 2. The `620 Application was filed on May 24, 1996, and issued as United States Patent No 5,748,547 ("the `547 Patent") on May 5, 1998. Ex 7.
- 3. The '290 Application was filed on February 25, 1997, as a continuation-in-part of the '620 Application, and issued as United States Patent No. 5,825,704 ("the '704 Patent") on October 20, 1998. Ex 9.
- 4. All three patents name Dr Jeng-Jye Shau as the sole inventor. Exs 7, 9 & 353.
- 5. Dr Shau's native language is Mandarin Chinese. Although he can communicate in English, Shau's English is imperfect. Doc #621 85:15 to 86:10. TSMC's expert stated that Shau's "patent

obviously is written by someone who doesn't have a great command of English." Doc #621 154:24 to 155:15.

- 6. Shau is an integrated circuit designer. Jury Trial 146:23-24.
- 7. The person of ordinary skill for purposes of Shau's patents is an integrated-circuit designer, and not an integrated-circuit manufacturer. Doc #621 134:4-9; 175:2 to 176:11. TSMC's own expert, Mr Taylor, defined a person of ordinary skill as having "a Bachelor's degree in electrical engineering and three to five years of experience in the design of semiconductor memory products, preferably DRAM products." Doc #621 134:7-9 (emphasis added). TSMC's expert therefore defined a person of ordinary skill with respect to experience and training in circuit design.
- 8. The work of an integrated-circuit designer is performed almost entirely on a computer. Circuit designers make extensive use of software-based circuit simulation, verification and layout tools. Doc #621 96:13 to 97:7; 101:9-12; 166:10-12; 177:9-22; 180:19 to 181:2.
- 9. The entire integrated-circuit design industry relies heavily on software simulation tools to predict accurately how a circuit will perform once manufactured. Doc #621 177:23 to 178:4; 188:5-18.
- 10. As a general practice, circuit designers simulate their designs extensively before having their circuit designs fabricated into a physical integrated-circuit, commonly called a "chip." Doc #621 188:9-16.
 - 11. A circuit designer's product is his design, and his or

her final product is an electronic design file known as a "tape out" file. Doc #621 181:16-20; 190:3-17. Even TSMC sometimes refers to design files as products. Doc #621 191:4-24. The tape out file serves as a blueprint from which the entirety of a design's details may be discerned and from which a circuit manufacturer can fabricate a chip. Jury Trial 177:8-12.

- 12. Once the design stages are completed including simulations, verifications, and the creation of a tape out file the design in the form of a tape out file is generally submitted to a circuit fabrication facility, commonly termed a "fab." Doc #621 181:12-15.
- 13. Whereas the final product of a circuit designer is a tape out and a circuit design, the final product of a fab, or foundry, is a physical chip.
- 14. Although circuit design and circuit manufacturing are part of the overall integrated-circuit production process, they are typically separate and distinct activities. Generally, circuit designers do not fabricate chips, and generally foundries do not design tape out files. Doc #621 181:16-24. Circuit designers and manufacturers by and large employ distinct vocabularies. Doc #621 177:1-5.
- 15. Integrated circuits from the point of view of a circuit-designer are under production when the circuit designer begins computer layout work on his or her design. This includes computer-simulations of the anticipated performance of the design, verification on a computer of the design's compliance with foundry-specific "design rules," creation of a tape-out file on a computer and continues until the tape out file is submitted to the

fab and sometimes continues therein and thereafter during an iterative refinement process. Doc #621 177:6 to 182:3.

- 16. UniRAM presented Dr Carl Sechen as an expert on how a circuit-designer with an appropriate background would interpret certain of Shau's patent statements that TSMC has challenged in the instant case. Dr Sechen holds a PhD from the University of California at Berkeley, is a fellow in the IEEE, has authored 150 publications in the field of integrated circuit design and has taught at Yale University, the University of Washington and the University of Texas at Dallas (where he is currently a full professor). Doc #621 170:13 to 172:21.
- 17. TSMC's expert on how one of ordinary skill would interpret certain of Shau's patent statements, Mr David Taylor, is less qualified than Dr Sechen. Mr Taylor lacks a PhD, is not a fellow in the IEEE and lacks the teaching and research experience of Dr Sechen. Ex 5291.
- 18. Mr Taylor, unlike Dr Sechen, cited no documents in either his expert report or his trial testimony to corroborate his opinions. Doc #621 152:3-20.
- 19. Dr Sechen consistently offered opinions that were more specific and complete than the answers given by Mr Taylor. Dr Sechen's opinions were supported by reference to objective standards in the industry, such as terms of art. Mr Taylor, by contrast, relied more often on his view of common sense and his personal interpretations of disputed phrases. Dr Sechen explained his interpretations of disputed terms in greater detail, usually by a more thorough consideration of the context in which the term appears.

2.7

20).	In May	1996,	Shau be	gan deve	eloping	an inte	egrated-cir	cuit
design	tha	t incl	uded a	dynamic	random	access	memory	("DRAM")	
archite	ectu	re. D	oc #62	1 88:22	to 89:2	•			

- 21. In May 1996, Shau began layout work on a tape out file using a software program named MAGIC and completed simulations using software programs named SPICE and RSIM. Doc #621 89:20 to 90:1; 90:10 to 91:3; Jury Trial 161:1 to 162:2.
- 22. The simulations reflected that the "access time" for Shau's DRAM architecture using 0.6 micron ("µm") technology design rules was 4 nanoseconds ("ns"). Doc #621 55:11-13; 72:17-24.
- 23. After confirming the viability of his DRAM architecture by simulation, and while he continued work on the tape out, Shau began drafting the '620 Application. Doc #621 91:11-19.
- 24. Shau is not a lawyer, and he drafted the '620 Application by himself, without assistance from counsel. Doc #621 88:11-21;
 Jury Trial 156:19-24.
- 25. The '620 Application as drafted included the following passage:

Our results show that a memory of the present invention is faster than an SRAM of the same memory capacity.

Ex. 5109, at 5109-014. The court finds this statement was not false or misleading.

26. At the time Shau made this statement, no physical "chip" had yet been fabricated. The '620 Application would not, however, suggest to a person of ordinary skill in the art that any physical semiconductor chips had been fabricated. Instead, this statement about "results" refers to simulations conducted on Shau's design.

Doc #621 202:11-18.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

27. Circuit designers simulate their designs regularly. Doc #621 94:5-10. UniRAM's expert stated that "[t]he whole [integrated circuit] industry relies extensively on simulation results to accurately predict the performance of the design." Doc #621 188:12-14. TSMC's expert stated that simulations provide "realistic predictions of just what the performance will be" once the design is fabricated. Doc #621 138:14-19.

- 28. From the perspective of a circuit designer, the term "results" in this sentence refers to simulation results that pertain to a design file, not results from testing a physical chip. Doc #621 93:20 to 94:10; 186:18 to 187:11.
- The challenged statement refers to "results" only, not "measured results." The choice of the word "results" is important because the phrase "measured results" is a term of art in the semiconductor industry. Doc #621 187:2-4. When persons of ordinary skill in the art discuss test results on a physical, fabricated chip as opposed to test results on a design, they often use the phrase "measured results." Doc #621 187:2-11. Creators of integrated circuits "get a lot of accolades for actually producing a part and having measured results." Doc #621 187:6-8. modifier "measured" in the phrase "measured results" is a very "prestigious and important qualifier" on the word "results." Doc #622 231:7-10. Accordingly, if and only if a circuit designer has a physical chip, he will use the term of art "measured results" to describe his test results. The absence of that term of art suggests strongly that the designer does not have a physical chip. Doc #621 187:9-11. Because Shau never in May 1996 made any

reference to "measured results," a person of ordinary skill would not read his statements to suggest the existence of a physical chip. Doc #621 186:18 to 187:11.

- 30. The Figures included in the `547 patent prove to one skilled in the art that the application discussed simulations and not actual chips. Figures 7a, 7b, and 7c in the `547 patent include artificial curves, input labels such as "BLKSEL" or "KWL" and output labels such as "BL" or "BL#" that a circuit designer would understand come from the SPICE simulation program and not a physical chip. Doc #621 91:9 to 93:19.
- 31. Circuit designers would also understand Figure 7a necessarily to relate to simulation results and not physical chip measurements. Figure 7a includes data on bitline outputs. It is not possible, however, to measure the bitline output of a physical chip. Accordingly, a circuit designer reviewing Figure 7a would know that the author of the patent had to have been reporting simulation results. Doc #621 98:15 to 99:4.
- 32. Similarly, the absence in the patent of any "chip photomicrographs" suggests that a person of ordinary skill in the art would infer that no physical chips had been produced. Doc #621 187:12 to 188:4. A chip photomicrograph is essentially a photograph that shows the details of the actual fabricated silicon chip. 187:16-22. Without such evidence, "no one would believe you have actually produced an actual chip." 187:23-24. Shau's patent did not include any reference to a chip photomicrograph. 187:25 to 188:4.
 - 33. The '620 Application includes the following passage:

Although the bit line structure in FIG. 3b is the actual bit line structure used in our product, for simplicity, we will use the simpler two-dimensional bit line structure in FIG. 3a as example in the following discussions.

Ex. 5109, at 5109-020. The court finds this statement was not false or misleading.

- 34. At the time Shau made this statement, no physical "chip" had yet been fabricated. A person of ordinary skill in the art, however, would also not interpret this statement to suggest that a physical "chip" had been manufactured. TSMC does not offer credible evidence to the contrary.
- 35. The audience of the patent is a circuit designer, and the subject matter of the patent is a circuit design. For circuit designers, the final product is a circuit design and a tape out file. As UniRAM's expert stated, because the person of ordinary skill in the art for the '620 Application is a circuit designer, the term "product" logically can mean a circuit design or a tape out file. Doc #621 190:3-17.
- 36. TSMC's expert stated that to a person of ordinary skill in the art, "product" refers to a chip that is "well beyond a test chip" and is "working to meet all specifications and is something that is available to be commercialized and sold." That testimony is less credible than the testimony of UniRAM's expert because the context and subject matter of the patent is not a physical chip that has been tested, but rather a design that has undergone simulations as described above.
- 37. There is no credible evidence that the statement "Although the bit line structure in FIG 3b is the actual bit line

2.7

structure used in our product, for simplicity, we will use the simpler two-dimensional bit line structure in FIG 3a as example in the following discussions" was false or misleading.

38. The '620 Application includes the following passage:

A memory device of the present invention is under production. Using 0.6 micron technology to build a memory array containing one million memory cells, we are able to achieve 4 ns access time, which is more than 10 times faster then existing memories devices of the same storage capacity.

Ex. 5109, at 5109-026. The court finds this statement was not false or misleading.

- 39. TSMC offers no credible evidence that the term "memory device" refers necessarily to physical fabricated chips. To the contrary, TSMC's expert stated that there is no "'nice, clear-cut totally clear definition of what a memory device is.'" Doc #621 154:6-23.
- 40. UniRAM's expert testified that in the context of the '229 patent, "memory device" refers to an integrated circuit which can exist as a design on a computer and is completely different from a "physical integrated circuit" or a "physical chip." #621 222:7-23.
- 41. UniRAM's expert stated that a memory device can exist "way before fabrication," including "in the design stage on a computer." #621 225:12-14. He specifically rejected the idea that a memory device refers to a physical chip. #621 222:16-23 and 220:10-11. In the context of the patent at issue, which discussed circuit design, it was "inconceivable" that the author was referring to a physical chip. Doc #622 226:4-10.

- 42. UniRAM's expert also testified that the phrase "device has been fabricated" is a term of art. Doc #621 183:16-20. Without that term of art, a person of ordinary skill in the art would be "convinced beyond any doubt that nothing has been actually fabricated." Doc #621 184:1-3. That term of art does not appear in the patent. Doc #621 183:16 to 184:3.
- 43. TSMC's only evidence that "memory device" refers necessarily to physical fabricated chips is its expert's statement that "I think in my view, it's pretty clear that when it says a device, it means something physical." Doc #621 152:24-25. He did not connect his interpretation to any terms of art or industry standards.
- 44. TSMC offers no credible evidence that the term "under production" means necessarily that physical chips are being fabricated for sale.
- 45. Unlike "under production," the phrase "in production" is a term of art in the semiconductor industry that "means that products intended for sale are actually being fabricated and produced." Doc #621 192:5-15; 182:22 to 183:4. UniRAM's expert compared having chips "in production" to producing automobiles on an assembly line in Detroit. Doc #622 215:6-24.
- 46. By contrast, the phrase "under production" does not have any widely understood specific meaning. Doc #621 183:1-4. In fact, because the phrase "in production" is a term of art referring specifically to mass production of chips, the use of the phrase "under production" suggests that physical chips are not being fabricated. Doc #622 215:6-24.

- 47. Because the meaning of "under production" is open-ended, the phrase should be interpreted in context. Doc #622 212:3-7. The context of the statement in the patent is circuit design, and thus the phrase "under production" refers to "the steps of the design process, including design, simulation, verification and on to the tapeout." Doc #621 183:5-12.
- 48. In addition, other assorted terms of art in the industry that also refer to the physical production of chips, such as "fabricated" or "manufactured," do not appear in the patent. Doc #621 183:16 to 184:3. The absence of any terms of art referring to physical production of chips suggests strongly that a person of ordinary skill in the art would not interpret "under production" as referring to the physical production of chips.
- 49. TSMC's expert testified that "under production" refers to the process of producing thousands of copies of a chip to test its viability before moving on to high-volume mass production, which runs in the millions of copies. Doc #621 139:25 to 141:16. His description of this pre-production process may be an accurate statement of production practices in the industry, but he offers no testimony that the specific term "under production" refers necessarily to the specific pre-production process he describes. He did not state that "under production" is a term of art, and he did not discuss whether his interpretation of "under production" is the same for the purposes of manufacturing processes and for the purposes of disclosures in a circuit design patent.
- 50. In the specific context of design disclosures in a circuit design patent, the use of the phrase "under production" to mean fabrication would be "highly unlikely." Doc #622 217:9-16.

This is because the usual practice for designers is to obtain a valid patent on a design <u>before</u> proceeding to manufacture and sale of the invention. Doc #622 215:19 to 216:15.

51. Because a circuit designer's final product is a circuit design and tape out, the statement in the '620 Application that a "memory device" was "under production" is consistent with a circuit design being in progress and the tape out being under production. Shau's design was "under production" as long as he was working on the layout of the chip design, including simulations. Doc #621 182:9 to 184:3; Jury Trial 269:1-19; 270:12-15. At the time the '620 Application was filed on May 24, 1996, most if not all of Shau's simulation work was finished and his layout work had begun. Doc #621 89:20 to 90:1. When he filed the '620 Application, Shau had completed the entire layout for the memory device that was claimed (but had not completed the external logic circuitry design). Doc #621 44:6-22. Shau's tape out, which is a circuit designer's product, was thus "under production" as of May 24, 1996.

- 52. The passage in the '620 Application "[u]sing 0.6 micron technology to build a memory array containing one million memory cells" referred to Shau's process of designing an array of a million memory cells and implementing the design with 0.6 micron technology design rules, not to a physical chip. Doc #621 95:9 to 96:8.
- 53. The reference to a "memory array" further shows that Shau was referring to a design and not a physical chip because an "array" is not a physical chip from a circuit designer's point of view. Doc #621 95:9 to 96:1; Jury Trial 271:8-15. Shau demonstrated a "memory array" for the jury on his computer -

navigating a software file, not a physical chip - while explaining his USRAM tape out. Jury Trial 188:20 to 189:3.

54. The reference to "build[ing]" an array also does not suggest necessarily a physical chip, because circuit designers - those of ordinary skill in the art - use the infinitive "to build" to discuss their work on design files, performed on a computer. Doc #621 96:13 to 97:10; 184:15 to 185:1; Jury Trial 271:19 to 272:8. The word "build" is not a term of art among circuit designers. Doc #621 184:21-22. Even if "build" refers commonly to physical structures, TSMC offers no evidence that "build" cannot be used as a metaphor for other purposes. According to Webster's Third International Dictionary, for instance, one can "build" an argument, a work of art or a piece of literature. Webster's Third New International Dictionary 291-92 (1981).

- 55. Shau's simulation results at the time of the '620 Application reflected a 4 ns access time, which was at relevant times more than ten times faster than traditional DRAM memories of the same size. Doc #621 94:2-17; 55:24 to 56:3; 97:18 to 98:3.
- 56. The reference to "4 ns" access time also would indicate to a person or ordinary skill in the art that Shau was referring to a design and not a physical chip because circuit designers use more than one digit of precision when referring to a physical, tested chip (for example, 4.x nanosecond access time) rather than a design (for example, 4 ns access time). Doc #621 185:25 to 186:17. Accordingly, Shau was not claiming to achieve a specific result, as he would if he were referring to a physical chip. TSMC's expert did not address Shau's use of one rather than two significant figures in his patent.

3	
4	

57.

5

6 7

8

10 11

12 13

14 15

16 17

18

19

20

21 22

23

24

25

26 27

art. Doc #621 55:24 to 56:2; 39:16 to 44:5. The reference to "one million memory cells" further indicates that a person of ordinary skill in the art would not read this patent language to suggest the presence of a physical chip. In the industry, memory cells in finished chips are always counted

Shau was not referring to Sunaga prior art when he

claimed that his invention was ten times faster than existing prior

Accordingly, a person of ordinary skill in the art would interpret Shau's imprecise description as suggesting that a design is being discussed. Doc #621 185:2-24.

as a power of two, and one million is not a power of two.

- 59. In the summer of 1996, Shau completed his first tape out and submitted it to MOSIS. Doc #621 102:10-15.
- MOSIS (not to be confused with MoSys, a former defendant in this case) is a relatively low-cost semiconductor manufacturing service often used for academic or research purposes. Doc #621 102:2-9.
- Shau realized that MOSIS could support only small test chips, so his original million memory cell array was reduced in size for the MOSIS tape out. Doc #621 107:23 to 108:14. Ex. 1378.
- 62. The MOSIS design was a test chip with a simple interface for testing purposes only. It was not configured as a commercial product. Doc #621 103:24 to 104:15.
- MOSIS sent Shau manufactured and packaged chips in late 1996. Doc #621 102:10-24.
- 64. Shau's initial tape out to MOSIS had bugs, but the bugs related to the logic portions of the tape out. The bugs did not relate to Shau's DRAM architecture, the subject of the claimed

inventions of Shau's patents. Shau was able to bypass the bugs to determine that the architecture of his design worked. Doc #621 102:10-21. The MOSIS tape out was successful. Doc #621 102:10-17.

- 65. Shau's success in verifying his memory architecture using the MOSIS prototype is reflected by his contemporaneous statement to TSMC in December 1996 that the "USRAM architecture has been verified on silicon by a 8K x 9 device." Ex 443 at 0041660.
- 66. Shau wished to verify his DRAM architecture on a full-size chip. Shau decided to approach TSMC, the world's largest independent commercial foundry, to begin commercial fabrication activities on his DRAM architecture designs. Doc #621 102:22 to 103:9.
- 67. In the latter part of 1996, pursuant to a nondisclosure agreement, Shau disclosed his DRAM architecture to TSMC and secured permission from TSMC to submit a tape out as a paying customer. Doc #621 103:10-17; Jury Trial 162:20 to 164:7.
- 68. In late 1996, Shau sent TSMC a tape out file. Shau's tape out to TSMC employed the same general DRAM architecture as the MOSIS tape out but was modified significantly. Doc #621 103:24 to 104:7. For example, the chip that Shau taped out to TSMC was much larger and had a more complex interface. Doc #621 104:2-7; 108:3-14.
- 69. Unlike his earlier MOSIS tape out, the design embodied in Shau's first tape out to TSMC was a test chip fully compatible with a popular commercial chip at the time, called CacheRAM. Doc #621 104:13-15; 106:16 to 107:8.

7	0.	TSN	IC suc	ccessi	Eully	CC	omple	eted	the	physi	cal	fabri	LCa	ation (of
Shau's	fi	rst	TSMC	tape	out	in	Janu	ıary	1997	and	sent	him	a	small	
number	of	tes	st chi	ips.	Jury	Tı	rial	211:	6 to	211:	24.				

- 71. Shau's initial TSMC tape out had bugs. However, the bugs were outside the portion of the tape out corresponding to Shau's DRAM architecture that is the subject of the claimed inventions of Shau's patents. Doc #621 109:17-25. Shau was able to bypass those errors for purposes of establishing that he could read and write data to the memory cells in the memory array. This meant that his memory array design was working properly. Doc #621 109:2 to 111:16; Jury Trial 211:25 to 213:1-18; 469:3-6; 214:17-22.
- 72. On February 25, 1997, Shau filed the '290 Application.

 The '290 Application included the following additional text in its specification:

Using this memory cell 1400 and a memory architecture disclosed in this invention and in our previous patent application, commercial memory products were manufactured successfully. The major advantage of the logic memory cell 1400 is that it can be manufactured using standard logic technology. The resulting memory product achieved unprecedented high performance.

Ex. 5132, at 5132-035.

- 73. As of the time of the filing of the `290 Application, physical chips corresponding to Shau's design had been fabricated successfully by TSMC. Doc #621 111:9-16; 116:21 to 117:2.
- 74. Because a physical semiconductor chip had been fabricated by February 1997, Shau included text in the '290 Application to reflect that a product had been "manufactured," whereas the term "manufactured" had not previously been employed in the '620

Application. Compare Ex 5109, with Ex 5132; see also Doc #621 183:23 to 184:9.

75. Because he had actually achieved (past tense) measured results from a manufactured chip, Shau used the term "achieved" in the '290 Application - rather than the present tense term "we are able to achieve" as was used in the '620 Application in reference to simulation results. Compare Ex 5109, at 5109-026, with Ex 5132, at 5132-035. The statement that the chip achieved unprecedented high performance was accurate. Doc #621 72:7; 72:22-24; 114:11-14.

76. The initial TSMC test chip was a "commercial" chip in the sense that it was configured as a commercial CacheRAM chip and was manufactured by a commercial foundry (TSMC) rather than an academic foundry (MOSIS). Doc #621 104:13-15; 106:16 to 107:8; 112:22-25. It was also a "commercial" chip in the sense that Shau intended it as a profit-making vehicle. Doc #621 49:19 to 50:15.

77. Most importantly, a person of ordinary skill in the art would consider a design to be a "commercial product" so long as it is intended and designed for eventual commercial sale - and not, for example, for experimental or academic purposes. Doc #621 194:18 to 195:3.

78. Based upon Shau's experience at Intel Corporation, he considered a product to be "commercial" if it was designed to be sold. Doc #621 112:24 to 113:5. Indeed, even the personnel at TSMC referred to Shau's USRAM product as a commercial product. Doc #621 113:6-8.

79. The phrase "manufactured successfully" is not a term of art, and its meaning is ambiguous rather than clear and unequivocal. Doc #621 195:4-10. Interpreted literally, the term

"manufactured successfully" in the '290 Application would indicate to a person of ordinary skill at a bare minimum that the circuit manufacturer had fabricated a chip successfully, the chip achieved a degree of functionality and the chip might lead to a successful commercial product. Doc #621 195:11 to 196:5.

- 80. The term "manufactured successfully" does not mean that the design is free of errors or bugs as long as the design is likely to lead to commercial success. Doc #621 195:16-21.
- 81. The first TSMC test chip had completed the manufacturing process, and, despite the presence of bugs in the extraneous logic portions of the chip, the memory portion of the chip (which contained the inventions claimed by the '290 Application) functioned successfully. Doc #621 109:2-25. Accordingly, the test chip was "manufactured successfully." Doc #621 114:2-5.
- **Manufactured successfully" in the '290 Application to reflect
 "full" or perfect functionality. The phrase "fully functional" is
 a term of art in the industry for perfect functionality, including
 functionality in areas of the chip that may have absolutely nothing
 to do with the claimed invention. Nor would a person of ordinary
 skill interpret "manufactured successfully" to suggest that the
 design was ready for mass production. Doc #621 195:4 to 196:22.
 Shau's statements to TSMC that his design was not worthy for mass
 production and not fully functional are consistent with his
 statement to the PTO that his claimed invention was successful; the
 claimed invention in his patent application related to the memory
 array (which worked perfectly), while the errors precluding mass
 production referred to the external logic circuitry.

8.	3.	In	the	lat	ter	half	of	1997,	Shau	ı beç	jan	design	ing	his	DRAI
archit	ectu	re	usir	ıg I	SMC	's 0.	35µ	n tech	nolog	gy•	Tha	t same	yea	ar,	Shau
submit	ted	to	TSMC	: a	tape	e out	de	signed	for	TSMO	C's	0.35µm	ı gei	neri	С
logic p	proc	ess		Jury	r Tri	ial 2	17:1	10-23.							

- 84. After revision of his 0.35µm TSMC tape out, Shau received fabricated chips from TSMC that were fully functional. Based on these fully functional chips, Shau filed the '538 Application, which ultimately issued as the '229 Patent. Jury Trial 217:10 to 218:20. Thus, a fully functional chip existed at the time the application for the '229 Patent was filed.
- 85. TSMC filed its motion for summary judgment on its inequitable conduct counterclaim on the eve of a mediation with UniRAM, suggesting that its pursuit of this claim was intended as settlement leverage against UniRAM. Doc #621 116:14-20.
- 86. In 2006, UniRAM reached a settlement with MoSys, a former defendant in this case. Doc #293.
- 87. To provide MoSys with complete relief, UniRAM had to include a patent release in the MoSys settlement that covered TSMC. Doc #287 at 3.
- 88. After executing the MoSys settlement, UniRAM pursued only trade-secret, and not patent, claims against TSMC. In November of 2006, UniRAM and TSMC filed jointly with the court a case management statement in which UniRAM stated that it was no longer seeking to enforce the '229 patent against TSMC. Doc #287 at 3.
- 89. Despite no longer needing to pursue its inequitable conduct theory as a defense against UniRAM's patent claims because UniRAM relinquished those claims TSMC did not drop its inequitable conduct counterclaim.

90. At the hearing on TSMC's motion for summary judgment,
counsel for TSMC informed the court that a motivation behind filing
the motion was to deny UniRAM's and UniRAM's counsel access to the
MoSys settlement funds: "UniRAM's a corporate entity and they could
pay [the settlement money] out to their shareholders. Whether
that's appropriate or not is a different question. There's
probably preexisting payments to law firms that need to be made.
They don't have a lot of money." Doc #381 (1/11/07 Inequitable
Conduct Summary Judgment Hearing Tr) at 3.

- 91. Relatedly, if TSMC had only been interested in protecting the settlement funds, it could have chosen a less drastic measure, such as an injunction, rather than seeking a judgment on its counterclaim. Doc #381 at 3-13. This suggests that TSMC's continued pursuit of its counterclaim was pretextual and was not in good faith.
- 92. In February 2007, the court denied TSMC's motion for summary judgment on inequitable conduct. Doc #349.
- 93. In the court's order denying TSMC's motion for summary judgment, the court indicated that TSMC had failed to identify any bases by which a PTO examiner could have made an obviousness or enablement rejection of the UniRAM patents. The court also labeled TSMC's interpretation of its evidence of Shau's intent to deceive the PTO as "odd." Doc #349 at 16.
- 94. Despite the summary judgment order that called into question the fundamental merits of TSMC's inequitable conduct counterclaim, TSMC continued to pursue that counterclaim after issuance of the order.

- 95. In continuing to pursue its inequitable conduct counterclaim after the summary judgment order, TSMC did not act to rectify the deficiencies in its position identified in that order. Doc #601 at 3-4.
- 96. UniRAM's trade secret claims proceeded to trial in September 2007.
- 97. During the trade secret trial, TSMC cross-examined Shau about most of the same supposed misstatements in his patent applications that form the basis of its inequitable conduct counterclaim. Jury Trial 265:25 to 287:5.
- 98. During the trade secret trial, TSMC's counsel argued to the jury in closing arguments that these supposed misstatements meant that Shau lacked credibility. Jury Trial 1841:15 to 1847:18.
- 99. Despite TSMC's arguments about the supposed misstatements in UniRAM's patents, the jury found in favor of Shau and UniRAM.

 Doc #544.
- 100. Even though TSMC's use of its supposed inequitable conduct evidence had not persuaded the jury to find against UniRAM, TSMC continued to pursue its inequitable conduct counterclaim against UniRAM following the jury verdict.

CONCLUSIONS OF LAW

A breach of the duties of candor, good faith, and honesty when prosecuting patent applications constitutes inequitable conduct. Molins PLC v Textron, Inc, 48 F3d 1172, 1178 (Fed Cir 1995). The proponent of an inequitable conduct defense has a "heavy burden to meet." UniRAM Tech, Inc v Taiwan Semiconductor

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

Mfg Co, No 04-1268 (VRW), 2007 WL 596397, at *2 (N D Cal Feb 21, 2007), citing Hoffman-La Roche, Inc v Promega Corp, 323 F3d 1354, 1359 (Fed Cir 2003).

- Inequitable conduct based upon an affirmative misstatement of fact or omission requires clear and convincing evidence of:
- a misrepresentation (by statement or omission) by the applicant;
 - b. the materiality of the misrepresentation and
 - intent to deceive the PTO.

Honeywell Intl Inc v Universal Avionics Sys Corp, 488 F3d 982, 999 (Fed Cir 2007); Syntex (USA) LLC v Apotex, Inc, 407 F3d 1371, 1384 (Fed Cir 2005) ("Materiality and intent to deceive are distinct factual inquiries, and each must be shown by clear and convincing evidence."), quoting Life Techs, Inc v Clontech Labs, Inc, 224 F3d 1320, 1324 (Fed Cir 2000).

- Once a material misstatement or omission and intent to deceive have been established, the district court must weigh these factors in light of all of the circumstances to determine whether a finding that inequitable conduct occurred is appropriate. Prods, Inc v Total Containment, Inc, 329 F3d 1358, 1362-63 (Fed Cir 2003).
- "[A] court must conduct a balancing test between the levels of materiality and intent, with a greater showing of one factor allowing a lesser showing of the other. Life Techs, 224 F3d at 1324.

2

3

4

5

6

7

8

10

11

13

14

15

16

17

18

19

20

21

22

23

24

25

26

- 5. Where the proponent of the inequitable conduct defense seeks to have a later patent declared unenforceable based upon alleged inequitable conduct in related patents, the proponent "must prove an 'immediate and necessary relation' between the inequitable conduct in the earlier patents and the enforcement of the descendent patent." <u>UniRAM</u>, 2007 WL 596397, at *2; see <u>Hoffman-La Roche</u>, Inc v Promega Corp, 319 F Supp 2d 1011 (N D Cal 2004).
- The fact that an applicant uses an imprecise term in a patent application does not provide "clear and convincing" evidence of a misstatement. Kothmann Enterprises, Inc v Trinity Indus, Inc, 455 F Supp 2d 608, 618-24 (S D Tex 2006), citing Hoffmann-La Roche, Inc v Promega Corp, 323 F3d 1354, 1363 (Fed Cir 2003), Purdue Pharma LP v Endo Pharms, Inc, 410 F3d 690 (Fed Cir 2005) and Frazier v Roessel Cine Photo Tech, Inc, 417 F3d 1230 (Fed Cir Although courts have found inequitable conduct based on "applicants' representations that they had performed experimental testing when they had not done so and that they had achieved test results that simply did not exist" (see Kothmann, 455 F Supp 2d at 623), here the evidence shows that Shau did perform tests. language Shau used to describe his claimed invention may at points have been imprecise. Shau's testimony describing the testing he did and the language he chose to use in his patent applications makes clear that the statements TSMC challenges were not misrepresentations. See Kothmann, 455 F Supp 2d at 623-24.
- 7. If the PTO needs more information in interpreting a patent application, the PTO examiner is authorized to seek clarification or additional information. Star Fruits SNC v United States, 393 F3d 1277, 1283 (Fed Cir 2005) ("The Office is clearly

entitled to use section 1.105 to seek information that may support a rejection. Just as the applicant produces information it deems pertinent to patentability under section 1.56, the examiner is free to request information under section 1.105 that the examiner deems pertinent to the issue of patentability.").

- 8. The PTO has adequate resources to investigate the representations made by applicants, if necessary. The PTO has the benefit of "hundreds of experts in the relevant arts to make independent inquiries." Aptix Corp v Quickturn Design Systems, Inc, 269 F3d 1369, 1379 (Fed Cir 2001) ("The courts have no greater resources to uncover fraud than the PTO. Although patent prosecutions are ex parte and judicial proceedings are adversarial, the PTO has the benefit of hundreds of experts in the relevant arts to make independent inquiries.") (Mayer dissenting in part).
- 9. Attorney fees may be awarded in an exceptional case. 35 USC § 285.
- 10. A finding that a case is exceptional imposes a more stringent requirement than the standard for proof of inequitable conduct, and in turn, an award of attorney fees imposes a more stringent requirement than the "exceptional case" standard. Argus Chemical Corp v Fibre Glass-Evercoat Co, 812 F2d 1381, 1387 (Fed Cir 1987) (Nies concurring).
- 11. Whether a case is "exceptional" is a question of fact.

 Brasseler, USA I, LP v Stryker Sales Corp, 267 F3d 1370, 1378 (Fed Cir 2001); Graco, Inc v Binks Mfg Co, 60 F3d 785, 794-95 (Fed Cir 1995) ("A finding by a court that a case is exceptional is a factual determination whereas the decision to award fees is discretionary.") (internal citations omitted). Direct or

circumstantial evidence that is clear and convincing is needed to establish an "exceptional case." <u>Brasseler</u>, 267 F3d at 1378-79.

- 12. Over-assertion of the defense of inequitable conduct has become an "absolute plague." <u>Burlington Indus, Inc v Dayco Corp</u>, 849 F2d 1418, 1422 (Fed Cir 1988).
- 13. Often litigants improperly assert the defense as a delay tactic, a tactic to obfuscate the issues before the court or to drive up the patentee's litigation costs. Chiron Corp v Abbott Labs, 156 FRD 219, 221 (N D Cal 1994).
- 14. "A patent litigant should be made to feel * * * that an unsupported charge of inequitable conduct in the Patent Office is a negative contribution to the rightful administration of justice."

 Burlington Indus, 849 F2d at 1422; see Fiskars, Inc v Hunt Mfg Co,

 221 F3d 1318, 1328 (Fed Cir 2000) (affirming an award of attorney fees for an inequitable conduct defense "so lacking in substance as to constitute a waste of the time and resources of all the participants").
- 15. Under 28 USC § 1927, courts have statutory authority to sanction attorneys for improper litigation conduct: "Any attorney * * * who so multiplies the proceedings in any case unreasonably and vexatiously may be required by the court to satisfy personally the excess costs, expenses, and attorneys' fees reasonably incurred because of such conduct."
- 16. Having reviewed the evidence and considered the testimony at trial, the court considers the testimony of UniRAM's witnesses to be more credible as to the existence or nonexistence of alleged misstatements in the patents and patent applications at issue in this proceeding than the evidence and testimony presented by TSMC.

- 17. The testimony of TSMC's primary witness as to falsity rested primarily upon his experience alone and was uncorroborated by any documents or other witnesses. Accordingly, it fails to provide clear and convincing evidence of falsity.
- 18. The basis of TSMC's claim is that some of Shau's statements can be possibly read in a context that would make the statements misleading. TSMC does not come close to proving that the alleged misstatements were false or misleading.
- 19. To the extent that the words used in the patents and patent applications at issue were imprecise, this imprecision does not give rise to clear and convincing evidence of falsity.
- 20. The court concludes that TSMC failed to prove by a preponderance of the evidence, much less clear and convincing evidence, that Shau made any misstatements in the patents and patent applications at issue.
- 21. Because TSMC has failed to establish by clear and convincing evidence the existence of any misstatements, the court concludes that there was no inequitable conduct in the prosecution of any of the patents at issue. The issues of materiality and intent are moot.
- 22. In light of the circumstances, the court finds that this is an exceptional case because TSMC pursued its counterclaim long after UniRAM dropped its patent infringement case. Moreover, circumstantial evidence suggests that TSMC brought this counterclaim not because it believed that Shau lied to the patent office but because it wanted to intimidate UniRAM. Most importantly, TSMC's evidence of falsity was barren. TSMC's pursuit of its inequitable conduct contention is highly questionable.

Nonetheless, the court declines to order TSMC to pay the attorney fees UniRAM incurred in defending the defense and counterclaim.

Awarding fees is an extraordinary sanction, and the Federal Circuit has not developed detailed criteria for such awards in the context of inequitable conduct claims.

VAUGHN R WALKER

United States District Chief Judge

IT IS SO ORDERED.

PATENT APPLICATION ATTY. DOCKET NO. 00100.02.0001

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

FILING OF A UNITED STATES PATENT APPLICATION

A GRAPHICS PROCESSING ARCHITECTURE EMPLOYING A UNIFIED SHADER

INVENTORS:

Steven Morein	Laurent Lefebvre					
10 Magazine, Apt. 801	124 Parenchere					
Cambridge, Massachusetts 02139	Lachenaie Quebec Canada J6W 6A5					
Andy Gruber 215 Pleasant Street Arlington, Massachusetts 02476	Andi Skende 49 Sheridan Drive, #11 Shrewsbury, Massachusetts 01545					

ASSIGNEE:

ATI Technologies, Inc.

1 Commerce Valley Drive East
Markham, Ontario
Canada L3T 7X6

ATTORNEY OF RECORD:
CHRISTOPHER J. RECKAMP
REGISTRATION NO. 34,414
VEDDER, PRICE, KAUFMAN & KAMMHOLZ, P.C.
222 NORTH LASALLE STREET, SUITE 2600
CHICAGO, ILLINOIS 60601
PHONE (312) 609-7500
FAX (312) 609-5005

Express Mail Label No. EL982266165US

Date of Deposit: November 20, 2003

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. Section 1.10 on the date of deposit, indicated above, and is addressed to: Mail Stop Patent Application, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450.

Name of Depositor: Margaret Caruso

Signature: Marquit Carus

ATI 2076 LG v. ATI IPR2015-00326 A GRAPHICS PROCESSING ARCHITECTURE EMPLOYING A UNIFIED SHADER

FIELD OF THE INVENTION

[0001] The present invention generally relates to graphics processors and, more

particularly, to a graphics processor architecture employing a single shader.

BACKGROUND OF THE INVENTION

[0002] In computer graphics applications, complex shapes and structures are formed

through the sampling, interconnection and rendering of more simple objects, referred to

as primitives. An example of such a primitive is a triangle, or other suitable polygon.

These primitives, in turn, are formed by the interconnection of individual pixels. Color

and texture are then applied to the individual pixels that comprise the shape based on

their location within the primitive and the primitives orientation with respect to the

generated shape; thereby generating the object that is rendered to a corresponding display

for subsequent viewing.

[0003] The interconnection of primitives and the application of color and textures to

generated shapes are generally performed by a graphics processor. Conventional

graphics processors include a series of shaders that specify how and with what

corresponding attributes, a final image is drawn on a screen, or suitable display device.

As illustrated in FIG. 1, a conventional shader 10 can be represented as a processing

block 12 that accepts a plurality of bits of input data, such as, for example, object shape

data (14) in object space (x,y,z); material properties of the object, such as color (16);

texture information (18); luminance information (20); and viewing angle information (22)

1

and provides output data (28) representing the object with texture and other appearance

properties applied thereto (x', y', z').

[0004] In exemplary fashion, as illustrated in FIGS. 2A-2B, the shader accepts the vertex

coordinate data representing cube 30 (FIG. 2A) as inputs and provides data representing,

for example, a perspectively corrected view of the cube 30' (FIG. 2B) as an output. The

corrected view may be provided, for example, by applying an appropriate transformation

matrix to the data representing the initial cube 30. More specifically, the representation

illustrated in FIG. 2B is provided by a vertex shader that accepts as inputs the data

representing, for example, vertices V_X, V_Y and V_Z, among others of cube 30 and

providing angularly oriented vertices $V_{X'}, V_{Y'}$ and $V_{Z'}$, including any appearance

attributes of corresponding cube 30'.

[0005] In addition to the vertex shader discussed above, a shader processing block that

operates on the pixel level, referred to as a pixel shader is also used when generating an

object for display. Generally, the pixel shader provides the color value associated with

each pixel of a rendered object. Conventionally, both the vertex shader and pixel shader

are separate components that are configured to perform only a single transformation or

operation. Thus, in order to perform a position and a texture transformation of an input,

at least two shading operations and hence, at least two shaders, need to be employed.

Conventional graphics processors require the use of both a vertex shader and a pixel

shader in order to generate an object. Because both types of shaders are required, known

graphics processors are relatively large in size, with most of the real estate being taken up

by the vertex and pixel shaders.

2

[0006] In addition to the real estate penalty associated with conventional graphics processors, there is also a corresponding performance penalty associated therewith. In conventional graphics processors, the vertex shader and the pixel shader are juxtaposed in a sequential, pipelined fashion, with the vertex shader being positioned before and operating on vertex data before the pixel shader can operate on individual pixel data.

[0007] Thus, there is a need for an improved graphics processor employing a shader that is both space efficient and computationally effective.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The present invention and the associated advantages and features thereof, will become better understood and appreciated upon review of the following detailed description of the invention, taken in conjunction with the following drawings, where like numerals represent like elements, in which:

[0009] FIG. 1 is a schematic block diagram of a conventional shader;

[00010] FIGS. 2A-2B are graphical representations of the operations performed by the shader illustrated in FIG. 1;

[00011] FIG. 3 is a schematic block diagram of a conventional graphics processor architecture;

[00012] FIG. 4A is a schematic block diagram of a graphics processor architecture according to the present invention;

[00013] FIG. 4B is a schematic block diagram of an optional input component to the graphics processor according to an alternate embodiment of the present invention; and

[00014] FIG. 5 is an exploded schematic block diagram of the unified shader employed in the graphics processor illustrated in FIG. 4A.

DETAILED DESCRIPTION OF THE INVENTION

[00015] Briefly stated, the present invention is directed to a graphics processor that employs a unified shader that is capable of performing both the vertex operations and the pixel operations in a space saving and computationally efficient manner. In an exemplary embodiment, a graphics processor according to the present invention includes an arbiter circuit for selecting one of a plurality of inputs for processing in response to a control signal; and a shader, coupled to the arbiter, operative to process the selected one of the plurality of inputs, the shader including means for performing vertex operations and pixel operations, and wherein the shader performs one of the vertex operations or pixel operations based on the selected one of the plurality of inputs.

[00016] The shader includes a general purpose register block for storing at least the plurality of selected inputs, a sequencer for storing logical and arithmetic instructions that are used to perform vertex and pixel manipulation operations and a processor capable of executing both floating point arithmetic and logical operations on the selected inputs according to the instructions maintained in the sequencer. The shader of the present invention is referred to as a "unified" shader because it is configured to perform both vertex and pixel operations. By employing the unified shader of the present invention,

the associated graphics processor is more space efficient than conventional graphics processors because the unified shader takes up less real estate than the conventional multi-shader processor architecture.

[00017] In addition, according to the present invention, the unified shader is more computationally efficient because it allows the shader to be flexibly allocated to pixels or vertices based on workload.

[00018] Referring now to FIG. 3, illustrated therein is a graphics processor incorporating a conventional pipeline architecture. As shown, the graphics processor 40 includes a vertex fetch block 42 which receives vertex information relating to a primitive to be rendered from an off-chip memory 55 on line 41. The fetched vertex data is then transmitted to a vertex cache 44 for storage on line 43. Upon request, the vertex data maintained in the vertex cache 44 is transmitted to a vertex shader 46 on line 45. As discussed above, an example of the information that is requested by and transmitted to the vertex shader 46 includes the object shape, material properties (e.g. color), texture information, and viewing angle. Generally, the vertex shader 46 is a programmable mechanism which applies a transformation position matrix to the input position information (obtained from the vertex cache 44), thereby providing data representing a perspectively corrected image of the object to be rendered, along with any texture or color coordinates thereof.

[00019] After performing the transformation operation, the data representing the transformed vertices are then provided to a vertex store 48 on line 47. The vertex store 48 then transmits the modified vertex information contained therein to a primitive

assembly block 50 on line 49. The primitive assembly block 50 assembles, or converts, the input vertex information into a plurality of primitives to be subsequently processed. Suitable methods of assembling the input vertex information into primitives is known in the art and will not be discussed in greater detail here. The assembled primitives are then transmitted to a rasterization engine 52, which converts the previously assembled primitives into pixel data through a process referred to as walking. The resulting pixel data is then transmitted to a pixel shader 54 on line 53.

The pixel shader 54 generates the color and additional appearance attributes that are to be applied to a given pixel, and applies the appearance attributes to the respective pixels. In addition, the pixel shader 54 is capable of fetching texture data from a texture map 57 as indexed by the pixel data from the rasterization engine 52 by transmitting such information on line 55 to the texture map. The requested texture data is then transmitted back from the texture map 57 on line 57' and stored in a texture cache 56 before being routed to the pixel shader on line 58. Once the texture data has been received, the pixel shader 54 then performs specified logical or arithmetic operations on the received texture data to generate the pixel color or other appearance attribute of interest. The generated pixel appearance attribute is then combined with a base color, as provided by the rasterization engine on line 53, to thereby provide a pixel color to the pixel corresponding at the position of interest. The pixel appearance attribute present on line 59 is then transmitted to post raster processing blocks (not shown).

[00021] As described above, the conventional graphics processor 40 requires the use of two separate shaders: a vertex shader 46 and a pixel shader 54. A drawback associated with such an architecture is that the overall footprint of the graphics processor

is relatively large as the two shaders take up a large amount of real estate. Another drawback associated with conventional graphics processor architectures is that can exhibit poor computational efficiency.

Referring now to FIG. 4A, in an exemplary embodiment, the graphics processor 60 of the present invention includes a multiplexer 66 having vertex (e.g. indices) data provided at a first input thereto and interpolated pixel parameter (e.g. position) data and attribute data from a rasterization engine 74 provided at a second input. A control signal generated by an arbiter 64 is transmitted to the multiplexer 66 on line 63. The arbiter 64 determines which of the two inputs to the multiplexer 66 is transmitted to a unified shader 62 for further processing. The arbitration scheme employed by the arbiter 64 is as follows: the vertex data on the first input of the multiplexer 66 is transmitted to the unified shader 62 on line 65 if there is enough resources available in the unified shader to operate on the vertex data; otherwise, the interpolated pixel parameter data present on the second input will be passed to the unified shader 62 for further processing.

[00023] Referring briefly to FIG. 5, the unified shader 62 will now be described. As illustrated, the unified shader 62 includes a general purpose register block 92, a plurality of source registers: including source register A 93, source register B 95, and source register C 97, a processor (e.g. CPU) 96 and a sequencer 99. The general purpose register block 92 includes sixty four registers, or available entries, for storing the information transmitted from the multiplexer 66 on line 65 or any other information to be maintained within the unified shader. The data present in the general purpose register block 92 is transmitted to the plurality of source registers via line 109.

[00024] The processor 96 may be comprised of a dedicated piece of hardware or can be configured as part of a general purpose computing device (i.e. personal computer). In an exemplary embodiment, the processor 96 is adapted to perform 32-bit floating point arithmetic operations as well as a complete series of logical operations on corresponding operands. As shown, the processor is logically partitioned into two sections. Section 96 is configured to execute, for example, the 32-bit floating point arithmetic operations of the unified shader. The second section, 96A, is configured to perform scaler operations (e.g. log, exponent, reciprocal square root) of the unified shader.

The sequencer 99 includes constants block 91 and an instruction store 98. [00025] The constants block 91 contains, for example, the several transformation matrices used in connection with vertex manipulation operations. The instruction store 98 contains the necessary instructions that are executed by the processor 96 in order to perform the respective arithmetic and logic operations on the data maintained in the general purpose register block 92 as provided by the source registers 93-95. The instruction store 98 further includes memory fetch instructions that, when executed, causes the unified shader 62 to fetch texture and other types of data, from memory 82 (FIG. 4A). In operation, the sequencer 99 determines whether the next instruction to be executed (from the instruction store 98) is an arithmetic or logical instruction or a memory (e.g. texture fetch) instruction. If the next instruction is a memory instruction or request, the sequencer 99 sends the request to a fetch block (not shown) which retrieves the required information from memory 82 (FIG. 4A). The retrieved information is then transmitted to the sequencer 99, through the vertex texture cache 68 (FIG. 4A) as described in greater detail below.

[00026] If the next instruction to be executed is an arithmetic or logical instruction, the sequencer 99 causes the appropriate operands to be transferred from the general purpose register block 92 into the appropriate source registers (93, 95, 97) for execution, and an appropriate signal is sent to the processor 96 on line 101 indicating what operation or series of operations are to be executed on the several operands present in the source registers. At this point, the processor 96 executes the instructions on the operands present in the source registers and provides the result on line 85. The information present on line 85 may be transmitted back to the general purpose register block 92 for storage, or transmitted to succeeding components of the graphics processor 60.

[00027] As discussed above, the instruction store 98 maintains both vertex manipulation instructions and pixel manipulation instructions. Therefore, the unified shader 99 of the present invention is able to perform both vertex and pixel operations, as well as execute memory fetch operations. As such, the unified shader 62 of the present invention is able to perform both the vertex shading and pixel shading operations on data in the context of a graphics controller based on information passed from the multiplexer. By being adapted to perform memory fetches, the unified shader of the present invention is able to perform additional processes that conventional vertex shaders cannot perform; while at the same time, perform pixel operations.

[00028] The unified shader 62 has ability to simultaneously perform vertex manipulation operations and pixel manipulation operations at various degrees of completion by being able to freely switch between such programs or instructions, maintained in the instruction store 98, very quickly. In application, vertex data to be processed is transmitted into the general purpose register block 92 from multiplexer 66.

The instruction store 98 then passes the corresponding control signals to the processor 96 on line 101 to perform such vertex operations. However, if the general purpose register block 92 does not have enough available space therein to store the incoming vertex data, such information will not be transmitted as the arbitration scheme of the arbiter 64 is not satisfied. In this manner, any pixel calculation operations that are to be, or are currently being, performed by the processor 96 are continued, based on the instructions maintained in the instruction store 98, until enough registers within the general purpose register block 92 become available. Thus, through the sharing of resources within the unified shader 62, processing of image data is enhanced as there is no down time associated with the processor 96.

[00029] Referring back to FIG. 4A, the graphics processor 60 further includes a cache block 70, including a parameter cache 70A and a position cache 70B which accepts the pixel based output of the unified shader 62 on line 85 and stores the respective pixel parameter and position information in the corresponding cache. The pixel information present in the cache block 70 is then transmitted to the primitive assembly block 72 on line 71. The primitive assembly block 72 is responsible for assembling the information transmitted thereto from the cache block 70 into a series of triangles, or other suitable primitives, for further processing. The assembled primitives are then transmitted on line 73 to rasterization engine block 74, where the transmitted primitives are then converted into individual pixel data information through a walking process, or any other suitable pixel generation process. The resulting pixel data from the rasterization engine block 74 is the interpolated pixel parameter data that is transmitted to the second input of the multiplexer 66 on line 75.

[00030] In those situations when vertex data is transmitted to the unified shader 62 through the multiplexer 66, the resulting vertex data generated by the processor 96, is transmitted to a render back end block 76 which converts the resulting vertex data into at least one of several formats suitable for later display on display device 84. For example, if a stained glass appearance effect is to be applied to an image, the information corresponding to such appearance effect is associated with the appropriate position data by the render back end 76. The information from the render back end 76 is then transmitted to memory 82 and a display controller line 80 via memory controller 78. Such appropriately formatted information is then transmitted on line 83 for presentation on display device 84.

[00031] Referring now to FIG. 4B, shown therein is a vertex block 61 which is used to provide the vertex information at the first input of the multiplexer 66 according to an alternate embodiment of the present invention. The vertex block 61 includes a vertex fetch block 61A which is responsible for retrieving vertex information from memory 82, if requested, and transmitting that vertex information into the vertex cache 61B. The information stored in the vertex cache 61B comprises the vertex information that is coupled to the first input of multiplexer 66.

[00032] As discussed above, the graphics processor 60 of the present invention incorporates a unified shader 62 which is capable of performing both vertex manipulation operations and pixel manipulation operations based on the instructions stored in the instruction store 98. In this fashion, the graphics processor 60 of the present invention takes up less real estate than conventional graphics processors as separate vertex shaders and pixel shaders are no longer required. In addition, as the unified shader 62 is capable

of alternating between performing vertex manipulation operations and pixel manipulation operations, graphics processing efficiency is enhanced as one type of data operations is not dependent upon another type of data operations. Therefore, any performance penalties experienced as a result of dependent operations in conventional graphics processors are overcome.

[00033] The above detailed description of the present invention and the examples described therein have been presented for the purposes of illustration and description. It is therefore contemplated that the present invention cover any and all modifications, variations and equivalents that fall within the spirit and scope of the basic underlying principles disclosed and claimed herein.

CLAIMS

What is claimed is:

1. A graphics processor, comprising:

an arbiter circuit for selecting one of a plurality of inputs in response to a control signal; and

a shader, coupled to the arbiter circuit, operative to process the selected one of the plurality of inputs, the shader including means for performing vertex operations and pixel operations, and performing one of the vertex operations or pixel operations based on the selected one of the plurality of inputs, wherein the shader provides a appearance attribute.

- 2. The graphics processor of claim 1, further including a vertex storage block for maintaining vertex information.
- 3. The graphics processor of claim 2, wherein the vertex storage block further includes a parameter cache operative to maintain appearance attribute data for a corresponding vertex and a position cache operative to maintain position data for a corresponding vertex.
- 4. The graphics processor of claim 1, wherein the appearance attribute is color, and the color is associated with a corresponding pixel when the selected one of the plurality inputs is pixel data.

- 5. The graphics processor of claim 1, wherein the appearance attribute is position, and the position attribute is associated with a corresponding vertex when the selected one of the plurality of inputs is vertex data.
- 6. The graphics processor of claim 5, wherein the appearance attribute is color, and the color attribute is associated with a corresponding pixel when the selected one of the plurality of inputs is pixel data.
- 7. The graphics processor of claim 5, wherein the appearance attribute is one of the following: color, lighting, texture, normal and position data.
 - 8. The graphics processor of claim 1, wherein the appearance value is depth.
- 9. The graphics processor of claim 1, wherein the selection circuit is a multiplexer, and the control signal is provided by an arbiter, wherein the arbiter is coupled to the multiplexer.
- 10. The graphics processor of claim 1, wherein the shader provides vertex position data and further including a primitive assembly block, coupled to the shader, operative to generate primitives in response to the vertex position data.

11. The graphics processor of claim 10, further including a raster engine, coupled to the primitive assembly block, operative to generate pixel parameter data in response to the assembled vertex data.

12. The graphics processor of claim 1, wherein the shader generates pixel color information in response to the selected one of the plurality of inputs.

13. The graphics processor of claim 1, wherein the shader includes a register block for maintaining the selected one of the plurality of inputs, a computation element operative to perform arithmetic and logical operations on the data maintained in the register block, and a sequencer for maintaining the instructions that are executed by the computation element.

14. The graphics processor of claim 1, wherein the shader further includes circuitry operative to access a memory.

- 15. A unified shader, comprising:
 - a general purpose register block for maintaining data;
 - a processor unit; and
- a sequencer, coupled to the general purpose register block and the processor unit, the sequencer maintaining instructions operative to cause the processor unit to execute vertex calculation and pixel calculation operations on selected data maintained in the general purpose register block.
- 16. The shader of claim 15, wherein the sequencer further includes circuitry operative to fetch data from a memory.
- 17. The shader of claim 15, further including a selection circuit operative to provide information to the general purpose block in response to a control signal.
- 18. The shader of claim 15, wherein the processor unit executes instructions that generate a pixel color in response to the selected one of the plurality of inputs.
- 19. The shader of claim 15, wherein the processor unit executes vertex calculations while the pixel calculations are still in progress.
- 20. The shader of claim 15, wherein the processor unit generates vertex position and appearance data in response to a selected one of the plurality of inputs.

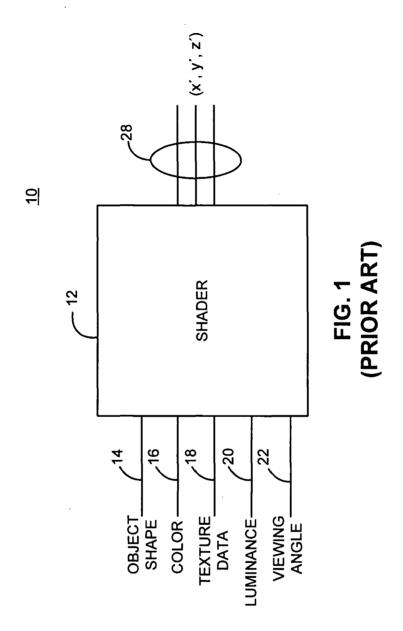
21. The shader of claim 17, wherein the selection circuit is a multiplexer and the control signal is provided by an arbiter.

A GRAPHICS PROCESSING ARCHITECTURE EMPLOYING A UNIFIED SHADER

ABSTRACT

A graphics processing architecture employing a single shader is disclosed. The architecture includes a circuit operative to select one of a plurality of inputs in response to a control signal; and a shader, coupled to the arbiter, operative to process the selected one of the plurality of inputs, the shader including means for performing vertex operations and pixel operations, and wherein the shader performs one of the vertex operations or pixel operations based on the selected one of the plurality of inputs. The shader includes a register block which is used to store the plurality of selected inputs, a sequencer which maintains vertex manipulation and pixel manipulations instructions and a processor capable of executing both floating point arithmetic and logical operations on the selected inputs in response to the instructions maintained in the sequencer.

CHICAGO/#837098.1



"A Graphics Processing Architecture Employing A Unified Shader" Inventors: Morein et al. Docket No. 0100.02.0001

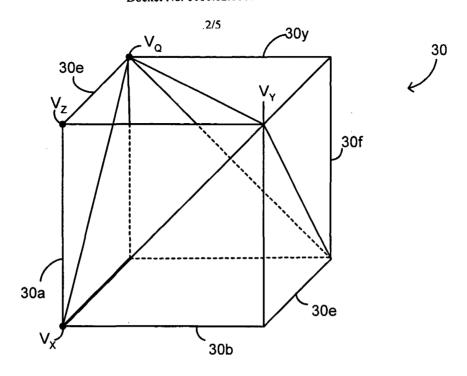


FIG. 2A (PRIOR ART)

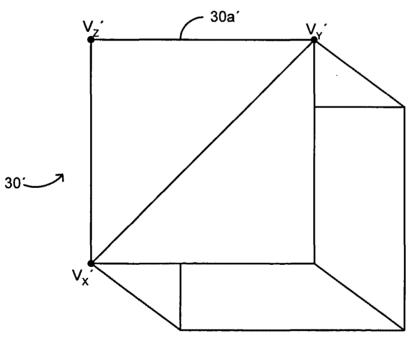
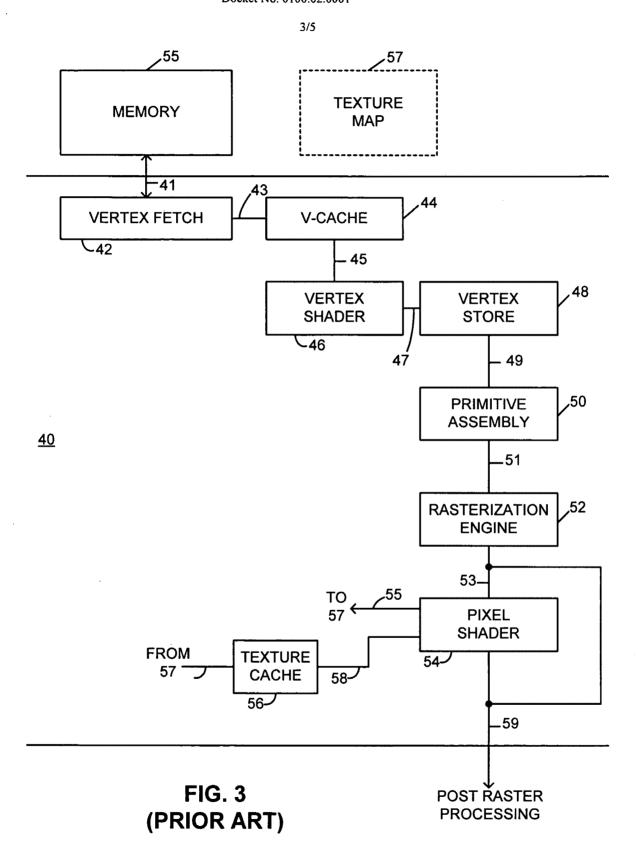


FIG. 2B (PRIOR ART)

"A Graphics Processing Architecture Employing A Unified Shader" Inventors: Morein et al. Docket No. 0100.02.0001



4/5

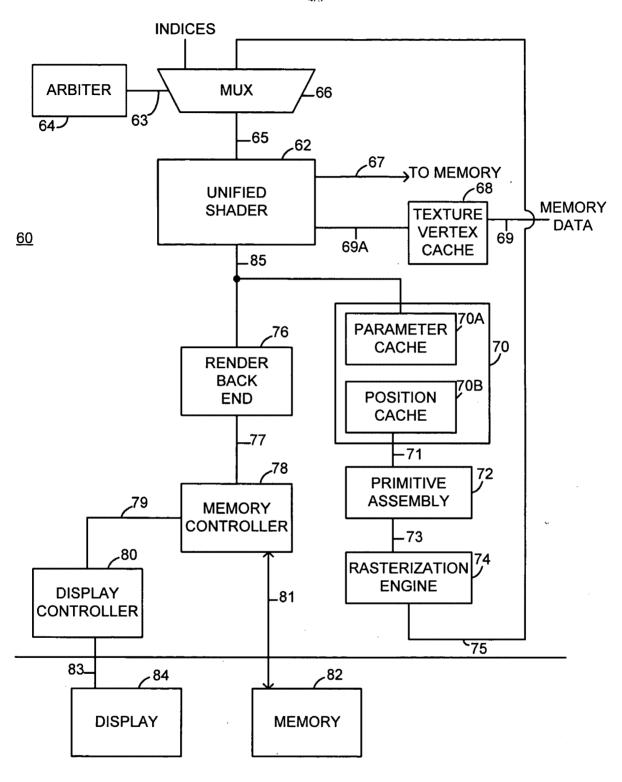
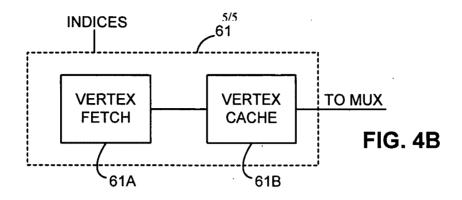
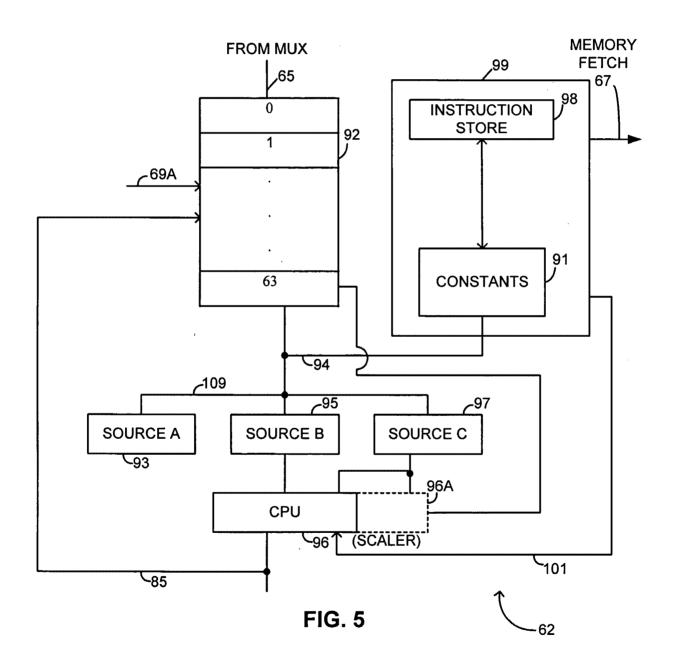
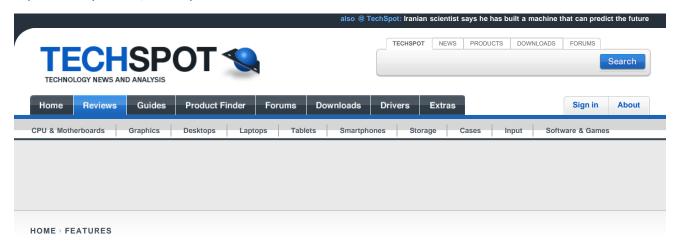


FIG. 4A

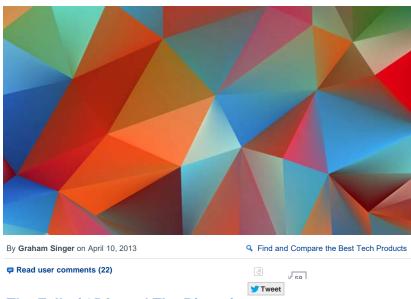
"A Graphics Processing Architecture Employing A Unified Shader" Inventors: Morein et al. Docket No. 0100.02.0001







History of the Modern Graphics Processor, Part 3



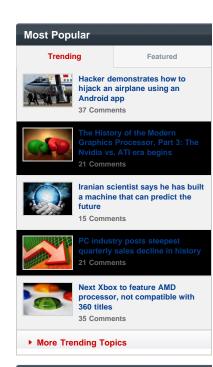


With the turn of the century the graphics industry bore witness to further consolidation.

The pro market saw iXMICRO leave graphics entirely, while NEC and Hewlett-Packard both produced their last products, the TE5 and VISUALIZE FX10 series respectively. Evans & Sutherland also parted ways with the sale of its RealVision line to focus on the planetaria and fulldome projection systems.

In the consumer graphics market, ATI announced the acquisition of ArtX Inc. in February 2000, for around \$400 million in stock. ArtX was developing the GPU codenamed Project Dolphin (eventually named "Flipper") for the Nintendo GameCube, which added significantly to ATI's bottom line.

Also in February, 3dfx announced a 20% workforce cut, then promptly moved to acquire Gigapixel for \$186



> Follow

TechSpot on:



 $https://web.archive.org/web/20130412060547/http://www.techspot.com/article/657-history-of-the-gpu-part-3/[9/12/2015\ 3:16:17\ PM]$

ATI 2077 LG v. ATI IPR2015-00326 million and gained the company's tile-based rendering IP

Meanwhile, S3 and Nvidia settled their outstanding patent suits and signed a seven-year cross-license agreement.



ATI GameCube GPU

VIA assumed control of S3 around April-May which itself

was just finishing a restructuring process from the acquisition of Number Nine. As part of S3's restructuring, the company merged with Diamond Multimedia in a stock swap valued at \$165 million. Diamond's high-end professional graphics division, FireGL, was spun off as SONICblue and later sold to ATI in March 2001 for \$10 million.

3DLabs acquired Intergraph's Intense3D in April, while the final acts of 3dfx played out towards the end of the year, despite 2000 kicking off with the promise of a better future as the long-awaited Voodoo 5 5500 neared its debut in July. The latter ended up trading blows with the GeForce 256 DDR and won the high-resolution battle.

Where 3dfx was once a byword for raw performance, its strengths around this time laid in its full screen antialiasing image quality.

But where 3dfx was once a byword for raw performance, its strengths around this time laid in its full screen antialiasing image quality. The Voodoo 5 introduced T-buffer technology as an alternative to transformation and lighting, by basically taking a few rendered frames and aggregating them into one image. This produced a slightly blurred picture that, when run in frame sequence, smoothed out the motion of the animation.

3dfx's technology became the forerunner of many image quality enhancements seen today, like soft

shadows and reflections, motion blur, as well as depth of field blurring.

3dfx's swan song, the Voodoo 4 4500, arrived October 19 after several delays – unlike the 4200 and 4800 that were never released. The card was originally scheduled for spring as a competitor to Nvidia's TNT2, but ended up going against the company's iconic GeForce 256 DDR instead, as well as the much better performing GeForce 2 GTS and ATI Radeon DDR.

On November 14, 3dfx announced they were belatedly ceasing production and sale of their own-branded graphics cards, something that had been rumoured for some time but largely discounted. Adding fuel to the fire, news got out that upcoming Pentium 4 motherboards would not support the 3.3V AGP signalling required Voodoo 5 series.



Legion Hardware Reviews

- HIS Radeon HD 7790 iCooler Turbo 1GB
- QNAP TS-469L
- Gigabyte GeForce GTX Titan
- HIS Radeon HD 7850 iPower IceQ Turbo
 4GB Crossfire
- Thecus N5550

Downloads and Drivers

Downloads

Drivers

- ♣ Speccy 1.21.491
- → 3DMark for Android 1.0.1-949
- UVK Ultra Virus Killer 5.3.0.0
- ♦ VLC Media Player 2.0.6
- LG PC Sync Suite
- More Downloads

Latest Discussion

- Speaker got dropped, now it gets no power
 - 7 replies on Audio and Video
- Gaming PC monitor, IPS or 120hz?
 - 2 replies on Audio and Video
- Used GeForce 9800GT 1GB overheats -should I buy it?
 - 14 replies on Audio and Video
- Graphic adapter for Dell XPS M1530?
 - 4 replies on Audio and Video
- Relation between fps and refresh rate14 replies on Audio and Video
- ▶ More Recent Discussion

Subscribe to TechSpot

Get free exclusive content, learn about new features and breaking tech news.





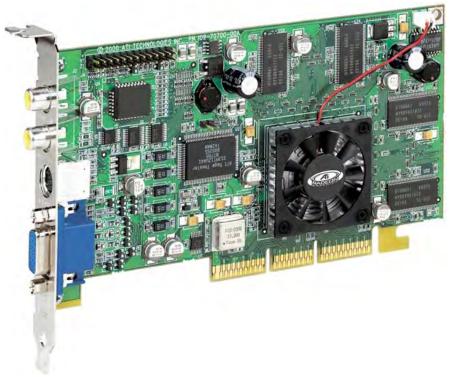
Voodoo5 5500 AGP box art

The death knell sounded a month later for 3dfx when Nvidia purchased its IP portfolio for \$70 million plus one million shares of common stock. A few internet wits later noted that the 3dfx design team which had moved to Nvidia eventually got both their revenge and lived up to their potential, by delivering the underperforming NV30 graphics chip powering the FX 5700 and FX 5800 cards behind schedule.

The Nvidia vs. ATI Era Begins

Prior to the Voodoo 5's arrival, ATI had announced the Radeon DDR as "the most powerful graphics processor ever designed for desktop PCs." Previews of the card had already gone public on April 25, and only twenty-four hours later Nvidia countered with the announcement of the GeForce 2 GTS (GigaTexel Shader). The latter included Nvidia's version of ATI's Pixel Tapestry Architecture, named Nvidia Shading Rasterizer, allowing for effects such as specular shading, volumetric explosion, refraction, waves, vertex blending, shadow volumes, bump mapping and elevation mapping to be applied on a per-pixel basis via hardware.

The feature was believed to have made it to the previous NV10 (GeForce 256) chip but it remained disabled due to a hardware fault. The GTS also followed ATI's Charisma Engine in allowing for all transform, clipping and lighting calculations to be supported by the GPU. That said, ATI went a step further with vertex skinning for a more fluid movement of polygons, and keyframe interpolation, where developers designed a starting and finishing mesh for an animation and the Charisma core calculated the intervening meshes.



ATI Radeon DDR

The ATI Radeon DDR eventually launched for retail in August 2000. Backed by a superior T&L implementation and support for several of the upcoming DirectX 8 features, the Radeon DDR alongside the GeForce 2 GTS ushered in the use of DVI outputs by integrating support for the interface into the chip itself. The DVI output was more often found on OEM cards, however, as the retail variety usually sported VIVO plugs.

One downside to the Radeon DDR is that boards shipped with their core and memory downclocked from the promised 200MHz and 183MHz, respectively. In addition, drivers were once again less than optimal at launch. There were issues with 16-bit color and compatibility problems with VIA chipsets, but this did not stop the card from dominating the competition at resolutions higher than 1024x768x32. A price of \$399 for the 64MB version stacked up well versus \$349-399 for the 64MB GeForce 2 GTS, which it beat by a margin of 10-20% in benchmarks, and helped ATI maintain its number one position in graphics market share over Nvidia.

Nvidia wasn't doing all that bad for themselves either. The company reported net income of \$98.5 million for the fiscal year on record revenue of \$735.3 million, driven in large part by its market segmentation strategy, releasing a watered-down MX version of the card in June and a higher clocked Ultra model in August. The latter dethroned the Radeon in terms of performance but it also cost \$499. A Pro model arrived in December.

Besides releasing a GeForce 2 card at every price point, from the budget MX to the professional Quadro 2 range, Nvidia also released its first mobile chip in the form of the GeForce2 Go.

As 3dfx was undergoing its death throes in November, Imagination Tech (ex-VideoLogic) and ST Micro attempted to address the high volume budget market with the PowerVR series 3 KYRO. Typically ranging in price from \$80 to \$110 depending on the memory framebuffer, the card represented good value for the money in gaming at resolutions of 1024x768 or lower. It would have become more popular, had the GeForce2 MX

By the time 2001 dawned, the PC graphics market consisted of a discrete card duopoly, with both

arrived later, or not so aggressively priced at ~\$110.

The KYRO II arrived in April 2001 with a bump in clock speeds compared to the original and manufactured on a smaller 180nm process by ST Micro. But once again the card faced stiff

of them in addition to Intel supplying the vast majority of integrated graphics chipsets.

competition from the GeForce 2 MX. Nvidia rebadged the card as the MX200 and lopped 40% off its price, while adding a higher clocked MX400 card at the same price as the Kyro II.

When PowerVR failed to secure game development impetus for tile based rendering, and ST Micro closed down its graphics business in early 2002, Imagination Technologies moved from desktop graphics to mobile and leveraged that expertise into system on chip graphics. They licenced the Series 5/5XT/6 for use with ARM-based processors in the ultra portable and smartphone markets.

By the time 2001 dawned, the PC graphics market consisted of a discrete card duopoly, with both of them in addition to Intel supplying the vast majority of integrated graphics chipsets.

Meanwhile, Matrox and S3/VIA clung to the margins of traditional markets.

Building on the strides made with the GeForce 2 series, Nvidia unveiled the GeForce 3 on February 27, 2001 priced between \$339 and \$449. The card became the new king of the hill, but it really only came into its own at the (then) extreme resolution of 1600x1200, preferably with full screen antialiasing applied.



Nvidia's stock GeForce 3 card

Initial drivers were buggy, especially in some OpenGL titles. What the new GeForce did bring to the table was DirectX 8, multisampling AA, quincunx AA (basically 2xMSAA + post process blur), 8x anisotrophic filtering as well as the unrivalled ability to handle 8xAF + trilinear filtering, and a programmable vertex shader which allowed for closer control of polygon mesh motion and a more fluid animation sequence.

There was also LMA (Lightspeed Memory Architecture) support -- basically Nvidia's version of HyperZ -- for culling pixels that would end up hidden behind others on screen (Z occlusion culling) as well as compressing and decompressing data to optimize use of bandwidth (Z compression).

Lastly, Nvidia implemented load-balancing algorithms as part of what they called the Crossbar Memory Controller, which consisted of four independent memory sub-controllers as opposed to the industry standard single controller, allowing incoming memory requests to be routed more effectively.



Nvidia NV2A inside Microsoft's Xbox

Nvidia's product line later added the NV2A, a derivative of the GeForce 3 with GeForce4 attributes that was used in Microsoft's Xbox game console.

At this point, Nvidia controlled 31% of the graphics market to Intel's 26% and ATI's 17%.

As Nvidia complemented the GF3 lineup with underclocked Ti 200 and overclocked Ti 500 models, ATI hurried to ramp up deliveries of the Radeon 8500. The card was built around the R200 GPU using TSMC's 150nm process (the same used by GeForce 3's NV20). The chip had been

announced in August and was eagerly awaited since John Carmack of id software talked it up saying it would run the new Doom 3 "twice as well" as the GeForce 3.

ATI's official R8500 announcement was no less enthusiastic. But reality kicked in once the card launched in October and was found to perform at the level of the underclocked GF3 Ti 200 in games. Unfinished drivers and a lack of workable Smoothvision antialiasing weighted heavily against the R8500 in its initial round of reviews. By the time the holiday season arrived, a second round of reviews showed that the drivers had matured to a degree and raised the R8500's performance in-between the Ti 200 and the standard GF3.

Spec comparison snapshot

	Core clock (MHz)	Pixel pipelines	Fill rate (Mpixels/s	Texture units per pixel pipeline	Fill rate (Mtexels/s)	Memory clock (MHz)	Memory bus width (bits)	Memory bandwidth (GB/s)
GeForce3 Ti 200	175	4	700	2	1400	400	128	6.4
GeForce3	200	4	800	2	1600	460	128	7.4
GeForce3 Ti 500	240	4	960	2	1920	500	128	8.0
Radeon 64MB DDR	183	2	366	3	1100	366	128	5.9
Radeon 8500	275	4	1100	2	2200	550	128	8.8

Very competitive pricing and a better all around feature set (2D image quality, video playback, performance under antialiasing) made the card a worthy competitor to the GF3 and Ti 500 nonetheless.

ATI's sales for the year dropped to \$1.04 billion as the company recorded a net loss of \$54.2 million. The company began granting licenses to board partners to build and market graphics boards, while refocusing their resources on design and chip making.

ATI also debuted the Set-Top-Wonder Xilleon, a development platform based on the Xilleon 220 SoC which provided a full processor, graphics, I/O, video and audio for set-top boxes integrated into digital TV designs.

To complement Xilleon, ATI acquired NxtWave

0006

https://web.archive.org/web/20130412060547/http://www.techspot.com/article/657-history-of-the-gpu-part-3/[9/12/2015~3:16:17~PM]



ATI Xilleon board

Communications for \$20 million in June 2002. The company specialized in digital signal processing and applications for set-top boxes and terrestrial

digital solutions.

Keeping up with their product launch cycle, Nvidia released the GeForce 4 in February 2002. Three MX parts, three mobile parts based on the MX models, and two performance Titanium models (Ti 4400 and Ti 4600) made up the initial line up -- built on TSMC's 150nm process. The GeForce 4 was effectively ready for release two months earlier but the launch was delayed to avoid eating into GeForce 3 sales over the holiday season.

The MX series cards were intended for the budget segment but they were still largely uninspiring as they were based on the old GeForce 2 architecture. MPEG2 decode added but the cards reverted to DirectX 7.0/7.1 support as the earlier GF2 MX line. Pricing at \$99-179 reflected the reduced feature set.

The Titanium models on the other hand were excellent performers and in some instances managed a 50+% increase in performance over the GeForce3 Ti 500. The Ti 4600 became the performance champ overnight, easily disposing of the Radeon 8500, while the Ti 4200 at \$199 represented the best value for money card.

But then came the Radeon 9700 Pro and promptly consigned every other card to also-ran status.



ATI Radeon 9700 Pro (FIC A97P)

Developed by a team that had originally formed the core of ArtX, the ATI R300 GPU delivered spectacularly and arrived very promptly. It was the first to bring DirectX 9.0 support, and by extension, the first architecture to support shader model 2.0, vertex shader 2.0, and pixel shader

2.0. Other notable achievements: it was the second GPU series to support AGP 8x -- SiS's Xabre 80/200/400 line was first -- and implementing the first flip-chip GPU package.

ATI complemented the line-up in October by adding a non-Pro 9700 at \$299 for those unable to part with \$399 for the top model. Meanwhile, the cut down 9500 Pro (\$199) and 9500 (\$179) reached down through mainstream market segments, and the FireGL Z1/X1 filled in the \$550-950 bracket for professional graphics. The All-In-Wonder 9700 Pro (\$449) was also added in December.

ATI's sales are likely to have taken a hit when it was found that many cards could be modded to their more expensive counterparts. Examples of this included the ability to turn a 9500 card into a 9700 using its reference board (with the full complement of memory traces), or a

About flip-chip GPU packages: Previous generations of graphics chips and other ICs used wire-bonding mounting. With this method, the chip sits on the board with the logic blocks sitting under the metal layers whose pads would be connected by thin wires arranged around the edges of the chip down to solder balls or pins on the underside. Flip-chip does away with the wire component through contact points (usually soldered in a ball grid array) directly on the "top" of the chip, which is then inverted, or "flipped" so that the solder points directly contact the substrate or circuit board. The chip then undergoes localised heating (reflow) to melt the solder that then forms the connection with the underlying contact points of the board.

9800 Pro to its XT counterpart. For the latter, a driver patch was made available to check if it would accept the mod, which consisted of soldering in a resistor or using a pencil to tweak the GPU and memory voltage control chip. Hard mods also included upgrading various 9800 models into a FireGL X2, while a patched/Omega driver had the ability to turn a \$250 9800 SE 256MB into a \$499 9800 Pro 256MB.

In addition to discrete graphics, ATI also introduced desktop integrated graphics and chipsets. These included the A3/ IGP 320 meant to be paired with AMD CPUs, RS200/IGP 330 & 340 for Intel chips, as well as the mobile series U1/IGP 320M for AMD platforms and RS200M for Pentium 4-M. All of them were complemented with ATI southbridges, specifically the IXP200/250.

SiS unveiled the Xabre line between the launch of the GeForce4 and the R300. The cards were consistently slower than Nvidia and ATI's offerings at the same price points, and were handicapped by the lack of vertex shader pipelines. This translated into a heavy reliance upon drivers and game developers to get the most out of software emulation, thus keeping SiS in the margins of desktop discrete 3D graphics.

The Xabre line also implemented "Turbo Texturing", where framerates were increased by drastically reducing texture quality, and lacked anisotrophic filtering. All this did little to endear reviewers to the cards.

The Xabre line was the last under the SiS banner, as the company spun off its graphics division (renamed XGI) and merged with Trident Graphics a couple of months later in June.

The first of Nvidia's FX series arrived on January 27, 2003 with the infamous "Dustbuster" FX 5800 and the slightly faster (read: less slow) FX 5800 Ultra. When compared to the reigning champ, the ATI Radeon 9700 Pro (and non-Pro), the FX was much louder, it delivered inferior anisotrophic filtering (AF) quality and antialiasing (AA) performance, and was overall much slower. ATI was so far ahead that a second-tier Radeon 9700 card launched five months earlier comfortably outperformed the Ultra, and it was \$100 cheaper (\$299 vs \$399).

The 3dfx design team which had moved to Nvidia got both their revenge and lived up to their potential, by delivering the underperforming NV30 graphics chip

behind schedule.

The NV30 chip was supposed to debut in August, around the same time as the Radeon 9700, but ramping problems and high defect rates on TSMC's Low-K 130nm process held Nvidia back. Some circles also argued that the company was strapped for engineering resources, with more than a few tied up with the NV2A Xbox console chip, the SoundStorm APU, as well as the motherboard chipsets.

Looking to move things forward Nvidia undertook a project to have several FX series chips fabricated on IBM's more conventional Fluorosilicate glass (FSG) low-K 130nm process.

ATI refreshed its line of cards in March, starting with the 9800 Pro, featuring a R350 GPU that was basically an R300 with some enhancements to the Hyper-Z caching and compression instruction.

The RV350 and RV280 followed in April. The first of these, found inside the Radeon 9600, was built using the same TSMC 130nm low-K process that Nvidia had adopted, Meanwhile, the RV280 powering the Radeon 9200 was little more than a rebadged RV250 of the Radeon 9000 with AGP 8x support.

The same month saw ATI and Nintendo sign a technology agreement that would eventually lead to the Hollywood GPU for the Nintendo Wii console. ATI added a second console coup in August, when Microsoft awarded the Xbox 360 GPU contract to them.

A scant three and a half months after the inglorious debut of the FX 5800, Nvidia took another shot with the NV35 (FX 5900 and FX 5900 Ultra). The new Detonator FX driver greatly improved AA and AF, almost matching ATI's solution in terms of quality. However the 5900 achieved what the 5800 could not. It knocked ATI's Radeon 9800 Pro from its spot as the fastest card around, although at \$499 apiece, few would actually take advantage of this.



Xbox 360 GPU (ATI C1 / Xenos)

As expected, ATI regained bragging rights in September with the release of the 9800 XT. Superior driver support – mainly with some DX9 games – also made the XT a better overall card than Nvidia's counterpart, ensuring that ATI ended the year with the performance crown. The 9700 Pro remained the standout mainstream board, while the FX 5700 Ultra at \$199 won the sub-\$200 price segment.

ATI bounced back with a \$35.2 million profit in 2003 after posting a \$47.5 million loss in 2002. A good chunk of this came from higher selling prices for the dominant 9800 and 9600 cards. Meanwhile, Nvidia retained 75% of the DirectX 9 value segment market, thanks to the popularity of the FX 5200.



0009



Source DirectX 9.0 Effects Trailer, shown during ATI's presentation of the Radeon 9800 XT and 9600 XT

The newly formed XGI launched the Xabre successor in a staggered release between September and November. Renamed Volari, the card line-up ranged from the \$49 V3 to the dual GPU Duo V8 Ultra. The V3 was virtually a rebrand of Trident's Blade XP4 and a DX 8.1 part, while the rest of the series (V5 and V8) was developed from the previous SiS Xabre and featured DX9.0 support.

For the most part, all of the models underdelivered, with the exception of the entry-level V3 which offered performance equal to the GeForce FX 5200 Ultra and Radeon 9200. The Duo V8 Ultra was priced ~20% higher than the Radeon 9800 Pro 128MB, yet delivered performance on par or lower than the 9600XT.

Another company making a comeback into desktop graphics was S3. Unfortunately, the buying public now generally saw desktop graphics as a two horse race – and S3 wasn't one of the two.

XGI's Volari line lingered on with the 8300 in late 2005, which was more or less on par with the Radeon X300SE/GeForce 6200 at \$49, as well as the Z9/Z11 and XP10. The company was reabsorbed back into SiS in October 2010.

Another company making a comeback into desktop graphics was S3. After the graphics division was sold to VIA for \$208 million plus the company's \$60 million debt, the restructured venture concentrated primarily on chipset projects.

DeltaChrome desktop cards were announced in January, but in time-honoured S3 fashion, the first S4 and S8 models didn't start appearing in the retail channel until December. The new cards featured most of the new must-haves of 2003;

DirectX 9 support, 16x AF, HD 1080p support, and portrait-mode display support.

Unfortunately, the buying public now generally saw desktop graphics as a two horse race – and S3 wasn't one of the two. While S3 was looking to keep competitive, ATI and Nvidia were driving each other to achieve ever-increasing levels of performance and image quality.

The DeltaChrome was succeeded by the GammaChrome in 2005.

Nvidia and ATI continued in 2005 their staggered launches. The former launched its first GDDR3 card in March as the FX 5700 Ultra, followed by the GeForce 6 series with the high-end 6800 range. The initial line up comprised the 6800 (\$299), GT (\$399), the Ultra (\$499), and an overclocked variant known as the Ultra Extreme (\$549) to counter ATI's X800 XT Platinum Edition. The latter was sold by a select band of add-in board partners.

The 6800 Ultra 512MB was added on March 14 2005 and sold for the unbelievable price of \$899 -- BFG added an overclocked version for \$999. The midrange was well catered for with the 6600 series in September.

Nvidia's feature set for the 6000 series included DirectX 9.0c support, shader model 3.0 (although the cards were never able to fully exploit this), Nvidia's PureVideo decode and playback engine,

and SLI support -- the multi-GPU performance multiplier IP that was acquired from 3dfx.



Reintroducing an old feature: SLI

Where the 3dfx implementation resulted in each processing unit being responsible for alternate line scans, Nvidia handled things in a few different ways. The company implemented split frame rendering (SFR), in which each GPU rendered the top or bottom half of the frame, alternate frame rendering (AFR) so GPUs rendered frames in turn, and in some cases the driver just disabled SLI depending on whether the game supported the feature. This last feature was a hit-or-miss early in driver development.

While the technology was announced in June, it required a motherboard with an nForce4 chipset to enable multi-GPU setups, and these didn't start reaching the retail channel in numbers until late November. Adding fuel to the fire, initial driver releases where sporadic (at best) until into the following year.

Reviews at the time generally mirrored current performance, showing that two lower tier cards (like the 6600 GT SLI which could be had for \$398) generally equalled one enthusiast card at lower resolutions and image quality. At highest resolutions and with antialiasing applied, however, single card setups still gained the upper hand. SLI and ATI's CrossFire performance was as erratic then as it sometimes is now, running the full gamut from perfect scaling to not working at all.

Nvidia's board partners immediately saw marketing opportunities with the re-invented tech, with Gigabyte offering a dual 6600 GT SLI card (the 3D1), followed by a dual 6600 (3D1-XL), and

While Nvidia's SLI was announced in June 2004, the required nForce4 motherboards didn't hit the retail channel in numbers until November, and initial driver releases where sporadic until

into the following year.

the 6800 GT (3D1-68GT). These cards not only required an nF4 chipset but also a Gigabyte branded motherboard as well.

Of the high-end single GPU cards, the 6800 Ultra and X800 XT/XT PE were fairly evenly matched, both in price and performance. But they weren't without their issues. The latter arrived in May and suffered supply constraints throughout its entire production life, while Nvidia's flagship 6800 Ultra was extremely late arriving in August and suffered supply constraints too depending on distribution area, since the card was only made available by a percentage of board partners.

The 6800 GT generally bested the X800 Pro at \$399, while the 6600 GT cleaned up in the \$199 bracket.

Intense competition with Nvidia that year didn't have an adverse effect on ATI's bottom line, as profit peaked at \$204.8 million for the year from nearly \$2 billion in revenue.

One quirk associated with the well-received 6600 GT was that it initially launched as a PCI Express card, at a time when PCI-E was an Intel-only feature for motherboards designed for Pentium 4 processors. These chips generally lagged in gaming performance behind AMD's offerings, which of course used the AGP data bus.

Nvidia's 7000 series started rolling off the assembly lines well before the 6000 series had completed its model line-up. The 7800 GTX arrived a full five months before the reduced bill of materials (BoM) 6800 GS saw the light of day. The first iteration of the 7800 series was based around the G70 GPU on TSMC's 110nm process, but quickly gave way to the G71-based 7900 series, made on TSMC's 90nm process.

While the naming convention changed from "NV" to "G", the latter were architecturally related to the NV40 series of the GeForce 6000. And while only fractionally larger than the NV40-45 at 334mm², the G70 packed in an extra eighty million transistors (for a total of 302 million), adding a third more vertex pipelines and 50% more pixel pipelines. In most cases, the G70 was superseded within nine months, and in the case of the GS and GTX 512MB, the figure was 3 and 4 months respectively.

At the entry level, the 7100 GS continued the use of TurboCache (the ability for the board to use some system memory), which was introduced with the previous generation GeForce 6200 TC.



Nvidia GeForce 7800 GTX

At the other end of the spectrum, the 7800 GTX 256MB hit retail on June 22 with an MSRP of \$599, though its actual street price was higher in many instances. ATI wrested the single-GPU crown back with the X1800 XT, but Nvidia countered with a 512MB version of the 7800 GTX thirty-five days later and promptly regained the title.

Two months later, ATI launched the X1900 XTX, which traded blows with Nvidia's flagship. This particular graphics horsepower race resulted in both cards being priced at \$650. One spinoff of the cards moving to a 512MB frame buffer was that gaming at 2560x1600 with 32-bit color and a high level of image quality enabled was now possible via dual link DVI.

ATI announced their multi-card Crossfire technology in May 2005 and made it available in September with the launch of the Xpress 200 Crossfire Edition chipset, and X850 XT Crossfire Master board. Due to a single-link TMDS, resolution and refresh rates were initially limited to 1600x1200 @60Hz, but a dual-link TMDS for 2560x1600 would soon replace it.



Unlike Nvidia's solution of two identical cards communicating via a bridge connector, ATI implemented a master card with TMDS receiver, which

ATI's original CrossFire design required using an external Y cable

accepted input from a slave card via external dongle and a Xilinx compositing chip.

Like Nvidia's SLI, CrossFire offered alternative frame rendering (AFR) and split frame rendering (SFR), but also a rendering technique called SuperTiling. The latter offered a performance increase in certain applications, but it did not work with OpenGL or support accelerated geometry processing. Also like SLI, Crossfire faced its share of driver-related troubles.

ATI intended to have their R520 based cards – their first to incorporate Shader Model 3.0 – ready by the June-July timeframe, but the late discovery of a bug in the cell library forced a 4 month delay.

Initial launches comprised the X1800 XL/XT using the R520 core, the X1300 budget cards using the RV515 with essentially one quarter of the graphics pipelines of the R520, and the X1600 Pro/XT based on the RV530, which was similar to the RV515 but with a higher shader and vertex pipeline-to-TMU and ROP ratio.

Due to the initial delay with the R520, the GPU and its derivations were being replaced a scant three and a half months later by the R580-based X1900 series which used TSMC's new 80nm process. Continuing with the roll out, half the graphics pipeline resources went into the RV570 (X1650 GT/XT and X1950 GT/Pro), while a shrunk RV530 became the RV535 powering the X1650 Pro as well as the X1300 XT.

ATI's revenue rose to a record \$2.2 billion for the year, the highest in the company's history, aided by shipments of Xenos GPUs for the Xbox 360. Net profit, however, slumped to \$16.9 million.

ATI's revenue rose to a record \$2.2 billion in 2005, the highest in the company's history, aided by shipments of Xenos GPUs for the Xbox 360. Net profit, however, slumped to \$16.9 million.

By this stage, any graphics card launch not based on an Nvidia or ATI GPU was received with a certain amount of curiosity, if not enthusiasm. Such was the scene when S3's overhauled graphics line-up debuted in November.

The Chrome S25 and S27 promised good gaming performance based on their high clocks, but delivered a mostly sub-par product. Initial pricing at \$99 (S25) and \$115 (S27) put the cards in competition against Nvidia's 6600/6600GT and ATI's X1300Pro/X1600Pro, but neither S3 card stood up to the competition in any meaningful way, aside from power consumption. That slight advantage evaporated as ATI/AMD and Nvidia

addressed the HTPC and entry-level market

segment, effectively killing S3's subsequent Chrome 400 and 500 series.

An added issue for S3 was that the cost of building the cards resulted in razor thin profits. The company needed high volume sales in a market dominated by two vendors. HTC were to acquire S3 in July 2012 for \$300 million, a move originally seen as leverage in HTC's and S3's separate legal disputes with Apple.

Nvidia and ATI continued to hog the press coverage in 2006.

ATI acquired Macrosynergy, a Shanghai based design and engineering centre with personnel working in California and previously part of the XGI group. Then in May the company bought BitBoys in a \$44 million deal.

Meanwhile, Nvidia's first foray into dual-GPU single board products came in March, following in the footsteps of ATI, 3dfx, and XGI. The 7900 GX2 would sandwich two custom boards essentially carrying a couple of downclocked 7900 GTXs. But Asustek didn't wait around for Nvidia's dual-GPU solution, however, and released its own take as the Extreme N7800GT Dual (\$900, 2000 units built), which paired two 7800 GT GPUs instead.

This card started Asus interest in limited edition dual-GPU boards, and possibly hardened Nvidia's attitude towards board partners', as Asustek products took the spotlight from their reference models at launch.

In the higher volume mainstream market, the 7600 GT and GS both provided solid performance and remarkable longevity, while ATI's X1950 XTX and Crossfire ruled the top end enthusiast benchmarks for single GPU cards. The X1900 XT and GeForce 7900 GT were fairly evenly matched in the upper mainstream bracket.



ATI's David Orton and AMD's Hector Ruiz officially announce the historic merger

After twenty-one years as an independent company, ATI was bought out by AMD on October 25 2006 for a total price of \$5.4 billion – split between \$1.7 billion from AMD, \$2.5 billion borrowed from lending institutions, 57 million AMD shares and 11 million options/restricted stock units valued at \$1.2 billion. At the time of the buy out, around 60-70% of ATI's chipset/IGP revenues were accrued from a partnership with Intel based motherboards.

Two weeks after the ATI buy-out, Nvidia ushered in

the age of unified shader architectures for PC graphics.

With a large part of Intel's IGP chipset market moving to Nvidia, market share dropped dramatically. The logic behind the buy was a seemingly quick path to GPU technology, rather than use the \$5.4 billion to develop AMD's own IP and add licenced technology where needed. At the time, AMD was aiming at the quick introduction of Torrenza and the associated Fusion projects.

Two weeks after the ATI buy-out, Nvidia ushered in the age of unified shader architectures for PC graphics. ATI's Xenos GPU for the Xbox 360 had already introduced the unified architecture to consoles.

This article is the third installment on a series of four. Next week we'll wrap things up, following the development of Radeon products under AMD's wing, the continued rivalry between GeForce and Radeon CPUs, the transition toward stream processing, and what the present a near future holds for graphics processors.

Part 1: (1976 - 1995) The Early Days of 3D Consumer Graphics

Part 2: (1995 - 1999) 3Dfx Voodoo: The Game-changer

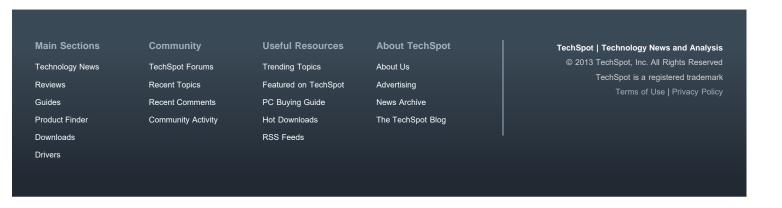
Part 3: (2000 - 2006) The Nvidia vs. ATI Era Begins

Part 4: (2006 - Present) The Modern GPU: Stream processing units a.k.a.

GPGPU

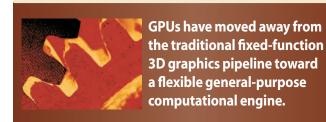
- 22 comments Read user comments, write your own
- Share this:
- Contact the author: Graham Singer
- Check out other recent features published at TechSpot





How GPUs Work

David Luebke, NVIDIA Research **Greg Humphreys**, University of Virginia



n the early 1990s, ubiquitous interactive 3D graphics was still the stuff of science fiction. By the end of the decade, nearly every new computer contained a graphics processing unit (GPU) dedicated to providing a high-performance, visually rich, interactive 3D experience.

This dramatic shift was the inevitable consequence of consumer demand for videogames, advances in manufacturing technology, and the exploitation of the inherent parallelism in the feed-forward graphics pipeline. Today, the raw computational power of a GPU dwarfs that of the most powerful CPU, and the gap is steadily widening.

Furthermore, GPUs have moved away from the traditional fixed-function 3D graphics pipeline toward a flexible general-purpose computational engine. Today, GPUs can implement many parallel algorithms directly using graphics hardware. Well-suited algorithms that leverage all the underlying computational horsepower often achieve tremendous speedups. Truly, the GPU is the first widely deployed commodity desktop parallel computer.

THE GRAPHICS PIPELINE

The task of any 3D graphics system is to synthesize an image from a description of a scene—60 times per second for real-time graphics such as videogames. This scene contains the geometric primitives to be viewed as well as descriptions of the lights illuminating the scene, the way that each object reflects light, and the viewer's position and orientation.

GPU designers traditionally have expressed this image-synthesis process as a hardware pipeline of specialized stages. Here, we provide a high-level overview of the classic graphics pipeline; our goal is to highlight those aspects of the real-time rendering calculation that allow graphics application developers to exploit modern GPUs as general-purpose parallel computation engines.

Pipeline input

Most real-time graphics systems assume that everything is made of triangles, and they first carve up any more complex shapes, such as quadrilaterals or curved surface patches, into triangles. The developer uses a computer graphics library (such as OpenGL or

Direct3D) to provide each triangle to the graphics pipeline one vertex at a time; the GPU assembles vertices into triangles as needed.

Model transformations

A GPU can specify each logical object in a scene in its own locally defined coordinate system, which is convenient for objects that are naturally defined hierarchically. This convenience comes at a price: before rendering, the GPU must first transform all objects into a common coordinate system. To ensure that triangles aren't warped or twisted into curved shapes, this transformation is limited to simple affine operations such as rotations, translations, scalings, and the like.

As the "Homogeneous Coordinates" sidebar explains, by representing each vertex in homogeneous coordinates, the graphics system can perform the entire hierarchy of transformations simultaneously with a single matrix-vector multiply. The need for efficient hardware to perform floating-point vector arithmetic for millions of vertices each second has helped drive the GPU parallel-computing revolution.

The output of this stage of the pipeline is a stream of triangles, all expressed in a common 3D coordinate system in which the viewer is located at the origin, and the direction of view is aligned with the *z*-axis.

Lighting

Once each triangle is in a global coordinate system, the GPU can compute its color based on the lights in the scene. As an example, we describe the calculations for a single-point light source (imagine a very small lightbulb). The GPU handles multiple lights by summing the contributions of each individual light. The traditional graphics pipeline supports the Phong lighting equation (B-T. Phong, "Illumination for Computer-Generated Images,' Comm. ACM, June 1975, pp. 311-317), a phenomenological appearance model that approximates the look of plastic. These materials combine a dull diffuse base with a shiny specular high-

96 Computer

ATI 2078 LG v. ATI IPR2015-00326 light. The Phong lighting equation gives the output color $C = K_d L_i(N \cdot L) + K_s L_i(R \cdot V)^s$.

Table 1 defines each term in the equation. The mathematics here isn't as important as the computation's structure; to evaluate this equation efficiently, GPUs must again operate directly on vectors. In this case, we repeatedly evaluate the dot product of two vectors, performing a four-component multiply-and-add operation.

Camera simulation

The graphics pipeline next projects each colored 3D triangle onto the virtual camera's film plane. Like the model transformations, the GPU does this using matrix-vector multiplication, again leveraging efficient vector operations in hardware. This stage's output is a stream of triangles in screen coordinates, ready to be turned into pixels.

Rasterization

Each visible screen-space triangle overlaps some pixels on the display; determining these pixels is called rasterization. GPU designers have incorporated many rasterizatiom algorithms over the years, which all exploit one crucial observation: Each pixel can be treated independently from all other pixels. Therefore, the machine can handle all pixels in parallel—indeed, some exotic machines have had a processor for each pixel. This inherent independence has led GPU designers to build increasingly parallel sets of pipelines.

Texturing

The actual color of each pixel can be taken directly from the lighting calculations, but for added realism, images called textures are often draped over the geometry to give the illusion of detail. GPUs store these textures in high-speed memory, which each pixel calculation must access to determine or modify that pixel's color.

In practice, the GPU might require multiple texture accesses per pixel to mitigate visual artifacts that can result when textures appear either smaller or larger on screen than their native

Homogeneous Coordinates

Points in three dimensions are typically represented as a triple (x,y,z). In computer graphics, however, it's frequently useful to add a fourth coordinate, w, to the point representation. To convert a point to this new representation, we set w = 1. To recover the original point, we apply the transformation $(x,y,z,w) \longrightarrow (x/w, y/w, z/w)$.

Although at first glance this might seem like needless complexity, it has several significant advantages. As a simple example, we can use the otherwise undefined point (x,y,z,0) to represent the direction vector (x,y,z). With this unified representation for points and vectors in place, we can also perform several useful transformations such as simple matrix-vector multiplies that would otherwise be impossible. For example, the multiplication

$$\begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

can accomplish translation by an amount Dx, Dy, Dz.

Furthermore, these matrices can encode useful nonlinear transformations such as perspective foreshortening.

resolution. Because the access pattern to texture memory is typically very regular (nearby pixels tend to access nearby texture image locations), specialized cache designs help hide the latency of memory accesses.

Hidden surfaces

In most scenes, some objects obscure other objects. If each pixel were simply written to display memory, the most recently submitted triangle would appear to be in front. Thus, correct hidden surface removal would require sorting all triangles from back to front for each view, an expensive operation that isn't even always possible for all scenes.

All modern GPUs provide a depth buffer, a region of memory that stores the distance from each pixel to the viewer. Before writing to the display, the GPU compares a pixel's distance to the distance of the pixel that's already present, and it updates the display memory only if the new pixel is closer.

THE GRAPHICS PIPELINE, EVOLVED

GPUs have evolved from a hardwired implementation of the graphics pipeline

to a programmable computational substrate that can support it. Fixed-function units for transforming vertices and texturing pixels have been subsumed by a unified grid of processors, or shaders, that can perform these tasks and much more. This evolution has taken place over several generations by gradually replacing individual pipeline stages with increasingly programmable units. For example, the NVIDIA GeForce 3, launched in February 2001, introduced programmable vertex shaders. These shaders provide units that the programmer can use for performing matrix-vector multiplication, exponentiation, and square root calculations, as

Table 1. Phong lighting equation terms.				
Term	Meaning			
K_d	Diffuse color			
$\overline{L_i}$	Light color			
N	Surface normal			
L	Vector to light			
K _s	Specular color			
R	Reflected light vector			
V	Vector to camera			
S	"Shininess"			

February 2007 **97**

HOW THINGS WORK



Figure 1. Programmable shading. The introduction of programmable shading in 2001 led to several visual effects not previously possible, such as this simulation of refractive chromatic dispersion for a "soap bubble" effect.

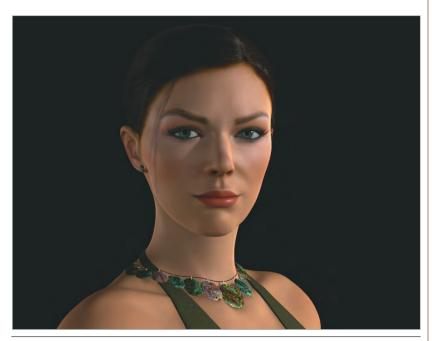


Figure 2. Unprecedented visual realism. Modern GPUs can use programmable shading to achieve near-cinematic realism, as this interactive demonstration shows, featuring actress Adrianne Curry on an NVIDIA GeForce 8800 GTX.

well as a short default program that uses these units to perform vertex transformation and lighting.

GeForce 3 also introduced limited reconfigurability into pixel processing,

exposing the texturing hardware's functionality as a set of *register combiners* that could achieve novel visual effects such as the "soap-bubble" look demonstrated in Figure 1. Subsequent

GPUs introduced increased flexibility, adding support for longer programs, more registers, and control-flow primitives such as branches, loops, and subroutines.

The ATI Radeon 9700 (July 2002) and NVIDIA GeForce FX (January 2003) replaced the often awkward register combiners with fully programmable pixel shaders. NVIDIA's latest chip, the GeForce 8800 (November 2006), adds programmability to the primitive assembly stage, allowing developers to control how they construct triangles from transformed vertices. As Figure 2 shows, modern GPUs achieve stunning visual realism.

Increases in precision have accompanied increases in programmability. The traditional graphics pipeline provided only 8-bit integers per color channel, allowing values ranging from 0 to 255. The ATI Radeon 9700 increased the representable range of color to 24-bit floating point, and NVIDIA's GeForce FX followed with both 16-bit and 32-bit floating point. Both vendors have announced plans to support 64-bit double-precision floating point in upcoming chips.

To keep up with the relentless demand for graphics performance, GPUs have aggressively embraced parallel design. GPUs have long used four-wide vector registers much like Intel's Streaming SIMD Extensions (SSE) instruction sets now provide on Intel CPUs. The number of such fourwide processors executing in parallel has increased as well, from only four on GeForce FX to 16 on GeForce 6800 (April 2004) to 24 on GeForce 7800 (May 2005). The GeForce 8800 actually includes 128 scalar shader processors that also run on a special shader clock at 2.5 times the clock rate (relative to pixel output) of former chips, so the computational performance might be considered equivalent to $128 \times 2.5/4 = 80$ four-wide pixel shaders.

UNIFIED SHADERS

The latest step in the evolution from hardwired pipeline to flexible computational fabric is the introduction of unified shaders. Unified shaders were first realized in the ATI Xenos chip for the Xbox 360 game console, and NVIDIA introduced them to PCs with the GeForce 8800 chip.

Instead of separate custom processors for vertex shaders, geometry shaders, and pixel shaders, a unified shader architecture provides one large grid of data-parallel floating-point processors general enough to run all these shader workloads. As Figure 3 shows, vertices, triangles, and pixels recirculate through the grid rather than flowing through a pipeline with stages of fixed width.

This configuration leads to better overall utilization because demand for the various shaders varies greatly between applications, and indeed even within a single frame of one application. For example, a videogame might begin an image by using large triangles to draw the sky and distant terrain. This quickly saturates the pixel shaders in a traditional pipeline, while leaving the vertex shaders mostly idle. One millisecond later, the game might use highly detailed geometry to draw intricate characters and objects. This behavior will swamp the vertex shaders and leave the pixel shaders mostly idle.

These dramatic oscillations in resource demands in a single image present a load-balancing nightmare for the game designer and can also vary unpredictably as the players' viewpoint and actions change. A unified shader architecture, on the other hand, can allocate a varying percentage of its pool of processors to each shader type.

For this example, a GeForce 8800 might use 90 percent of its 128 processors as pixel shaders and 10 percent as vertex shaders while drawing the sky, then reverse that ratio when drawing a distant character's geometry. The net result is a flexible parallel architecture that improves GPU utilization and provides much greater flexibility for game designers.

GPGPU

The highly parallel workload of real-time computer graphics demands

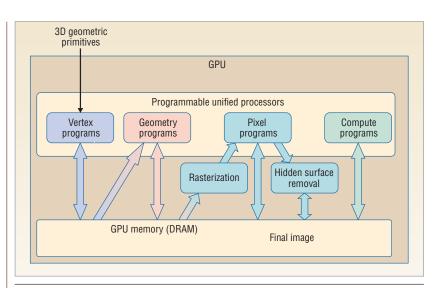


Figure 3. Graphics pipeline evolution. The NVIDIA GeForce 8800 GPU replaces the traditional graphics pipeline with a unified shader architecture in which vertices, triangles, and pixels recirculate through a set of programmable processors. The flexibility and computational power of these processors invites their use for general-purpose computing tasks.

extremely high arithmetic throughput and streaming memory bandwidth but tolerates considerable latency in an individual computation since final images are only displayed every 16 milliseconds. These workload characteristics have shaped the underlying GPU architecture: Whereas CPUs are optimized for low latency, GPUs are optimized for high throughput.

The raw computational horsepower of GPUs is staggering: A single GeForce 8800 chip achieves a sustained 330 billion floating-point operations per second (Gflops) on simple benchmarks (http://graphics.stanford.edu/projects/gpubench). The ever-increasing power, programmability, and precision of GPUs have motivated a great deal of research on general-purpose computation on graphics hardware—GPGPU for short. GPGPU researchers and developers use the GPU as a computational coprocessor rather than as an image-synthesis device.

The GPU's specialized architecture isn't well suited to every algorithm. Many applications are inherently serial and are characterized by incoherent and unpredictable memory access. Nonetheless, many important problems require significant computational

resources, mapping well to the GPU's many-core arithmetic intensity, or they require streaming through large quantities of data, mapping well to the GPU's streaming memory subsystem.

Porting a judiciously chosen algorithm to the GPU often produces speedups of five to 20 times over mature, optimized CPU codes running on state-of-the-art CPUs, and speedups of more than 100 times have been reported for some algorithms that map especially well.

Notable GPGPU success stories include Stanford University's Folding@ home project, which uses spare cycles that users around the world donate to study protein folding (http://folding.stanford.edu). A new GPU-accelerated Folding@home client contributed 28,000 Gflops in the month after its October 2006 release—more than 18 percent of the total Gflops that CPU clients contributed running on Microsoft Windows since October 2000.

In another GPGPU success story, researchers at the University of North Carolina and Microsoft used GPU-based code to win the 2006 Indy PennySort category of the TeraSort competition, a sorting benchmark testing price/performance for database

HOW THINGS WORK

operations (http://gamma.cs.unc.edu/ GPUTERASORT). Closer to home for the GPU business, the HavokFX product uses GPGPU techniques to accelerate tenfold the physics calculations used to add realistic behavior to objects in computer games (www. havok.com).

odern GPUs could be seen as the first generation of commodity data-parallel processors. Their tremendous computational capacity and rapid growth curve, far outstripping traditional CPUs, highlight the advantages of domain-specialized data-parallel computing.

We can expect increased programmability and generality from future

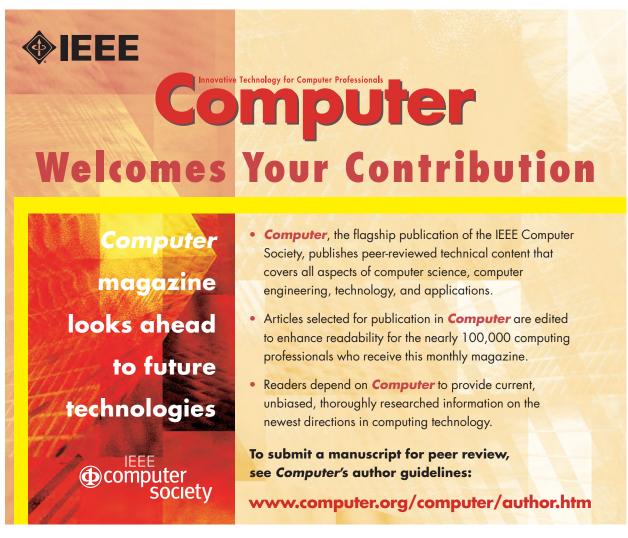
GPU architectures, but not without limit; neither vendors nor users want to sacrifice the specialized architecture that made GPUs successful in the first place. Today, GPU developers need new high-level programming models for massively multithreaded parallel computation, a problem soon to impact multicore CPU vendors as well.

Can GPU vendors, graphics developers, and the GPGPU research community build on their success with commodity parallel computing to transcend their computer graphics roots and develop the computational idioms, techniques, and frameworks for the desktop parallel computing environment of the future?

David Luebke is a research scientist at NVIDIA Research. Contact him at dluebke@nvidia.com.

Greg Humphreys is a faculty member in the Computer Science Department at the University of Virginia. Contact him at humper@cs.virginia.edu.

Computer welcomes your submissions to this bimonthly column. For additional information, or to suggest topics that you would like to see explained, contact column editor Alf Weaver at weaver@cs. virginia.edu.





News Center

Menu _■ N

Microsoft and ATI Technologies Announce Technology Development Agreement

REDMOND, Wash., Aug. 14, 2003 — Microsoft Corp. (Nasdaq

"MSFT") today announced it has entered into a technology development agreement with ATI Technologies Inc. (TSX: ATY, NASDAQ: ATYT). Under the agreement, ATI is developing custom, leading-edge graphics technologies for use in future Xbox® products and services.

"We're combining Microsoft's vision, software experience and R & D resources with ATI's pioneering leadership in graphics technologies to create innovative future Xbox products and services that meet the lifestyle needs of consumers in the Digital Decade."

said Robbie Bach, senior vice president of the Home and Entertainment Division at Microsoft.

"We selected ATI after reviewing the top graphics technologies in development and determining that ATI's technical vision fits perfectly with the future direction of Xbox."

"Microsoft shares our passion for cutting-edge innovation,"

said K. Y. Ho, chairman and chief executive officer, ATI Technologies Inc.

"Our success working with Microsoft in the past gives us great confidence as we move forward, and our broad experience and wealth of engineering resources will ensure that we deliver. This agreement cements ATI's position as the prime graphics supplier for the future of the games industry."

About ATI Technologies Inc.

ATI Technologies Inc. is a world leader in the design and manufacture of innovative 3-D graphics and digital media silicon solutions. An industry pioneer since 1985, ATI is the world's foremost visual processor unit (VPU) provider and is dedicated to delivering leading-edge performance solutions for the full range of PC and Mac desktop and notebook platforms, workstation, set-top and digital television, game console, and handheld markets. With 2002 revenues in excess of U.S. \$1 billion, ATI has more than 2,000 employees in the Americas, Europe and Asia. ATI common shares trade on Nasdaq (ATYT) and the Toronto Stock Exchange (ATY).

 $http://news.microsoft.com/2003/08/14/microsoft-and-ati-technologies-announce-technology-development-agreement/[9/12/2015\ 3:25:16\ PM]$

ATI 2079 LG v. ATI IPR2015-00326

About Xbox

Xbox (http://www.xbox.com/) is Microsoft's future-generation video game system that delivers the most powerful games experiences ever. Xbox empowers game artists by giving them the technology to fulfill their creative visions as never before, creating games that blur the lines between fantasy and reality. Xbox is now available in the continents of North America, Europe, Asia and Australia.

About Microsoft

Founded in 1975, Microsoft (Nasdag

"MSFT"

) is the worldwide leader in software, services and Internet technologies for personal and business computing. The company offers a wide range of products and services designed to empower people through great software — any time, any place and on any device.

Microsoft and Xbox are either registered trademarks or trademarks of Microsoft Corp. in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Note to editors: If you are interested in viewing additional information on Microsoft, please visit the Microsoft® Web page at http://www.microsoft.com/presspass/ on Microsoft's corporate information pages. Web links, telephone numbers and titles were correct at time of publication, but may since have changed. For additional assistance, journalists and analysts may contact Microsoft's Rapid Response Team or other appropriate contacts listed at http://www.microsoft.com/presspass/contactpr.asp.

Featured Posts

Microsoft and Dell partner to deliver Windows 10 devices and unmatched sales, services and support to the enterprise

ROUND ROCK, TX and REDMOND, WA - Sept. 8, 2015 - ... Read more »

Microsoft reinvents productivity with upcoming release of customer engagement solution REDMOND, Wash. — Sept. 8, 2015 — Microsoft Corp. (Nasdaq ... Read more »

Xbox unveils more of its greatest games lineup in history

COLOGNE, Germany — Aug. 4, 2015 — Today Xbox opened gamescom ... Read more »

Most Popular

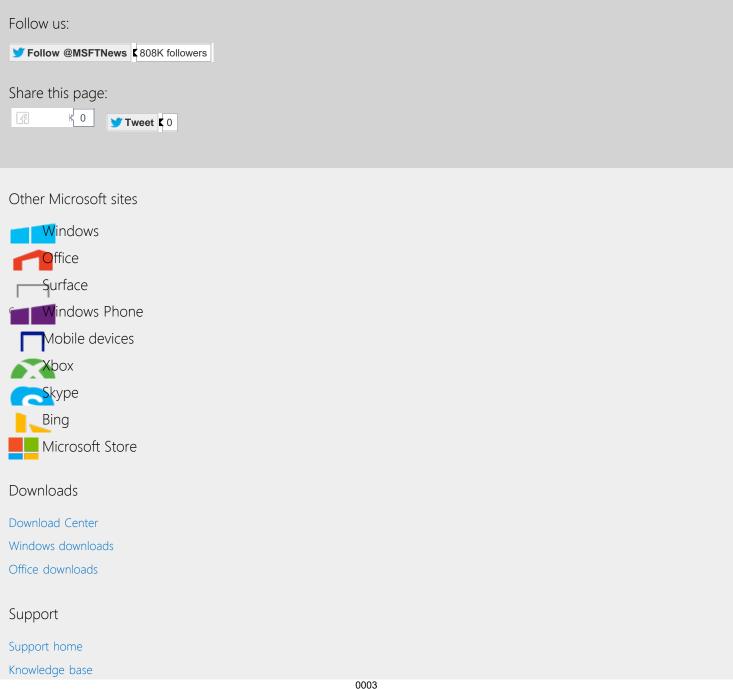
0002

http://news.microsoft.com/2003/08/14/microsoft-and-ati-technologies-announce-technology-development-agreement/[9/12/2015 3:25:16 PM]

Microsoft and Dell partner to deliver Windows 10 devices and unmatched sales, services and support to the enterprise

Microsoft reinvents productivity with upcoming release of customer engagement solution

Windows 10 Upgrade Your World Initiative: Beagle Freedom Project wins; more chances to vote for your favorite nonprofits in 10 countries



Security
Security home
Microsoft Security Essentials
About
Microsoft
Careers
Company News
Microsoft Account
Investor relations
Site map
Popular resources
Windows Phone devices
Windows Phone app and games
Laptops and desktop computers
Microsoft computer security
Malware removal tool
Malware removal tool Free antivirus software
Malware removal tool Free antivirus software Cloud computing solutions
Malware removal tool Free antivirus software
Malware removal tool Free antivirus software Cloud computing solutions
Malware removal tool Free antivirus software Cloud computing solutions

Microsoft© 2015 Microsoft



Advanced search





Once again X-bit labs asks you what you think about the companies participating in the hardware market and their results in 2004. Who is the leader and who is the follower? What is really fast and what is a marketing gimmick? Share your opinion with us - choose the best of the best in 2004.



ATI and NVIDIA Proclaim Different Graphics Processors Architecture Goals

ATI Says Unified Rendering Engine – the Way to Go, NVIDIA Disagrees

Video News Archive

by <u>Anton Shilov</u> 12/23/2004 | 07:55 AM

While particular approaches in graphics processing units design have been pretty different for leading computer visual companies ATI Technologies and NVIDIA Corp., in future the architecture of GPUs from the firms may be fundamentally different, as executives from both companies proclaim different approaches for chip internal architectures.

NVIDIA Disagrees with ATI Technologies

ATI Technologies' developer relations manager Richard Huddy said last month during a conference in London, UK, that the company's future visual processing units will feature unified pixel and shader processing. While he declined to elaborate on the timeframes for such chips, he said unified pixel and vertex data processing is a required capability for Windows Graphics Foundation 2.0 that comes out together with Microsoft's next-generation operating system called Windows Longhorn. On of the benefits the unified approach brings is ability to dynamically allocate chip resources depending on the demand for pixel and vertex processing, Mr. Huddy said. Another one is simplified software development.

NVIDIA Corp.'s chief architect David Kirk called the unified graphics engines as an implementation detail, not a feature, but admitted the unified architecture would be nice for programmers, who would have one instruction set for vertex and pixel shaders.

"It's not clear to me that an architecture for a good, efficient, and fast vertex shader is the same as the architecture for a good and fast pixel shader. A pixel shader would need far, far more texture math performance and read bandwidth than an optimized vertex shader. So, if you used that pixel shader to do vertex shading, most of the hardware would be idle, most of the time. Which is better – a lean and mean optimized vertex shader and a lean and mean optimized pixel shader or two less-efficient hybrid shaders? There is an old saying: 'Jack of all trades, master of none'," Mr. Kirk said in an interview with ExtremeTech web-site.

ATI: Bridging Today and Tomorrow

Not much is known about the architecture and capabilities of the codenamed R520 product scheduled for release in Q2 2005 that was initially referred as the R500. What is clear now is that the new graphics chip will

 Video

 December, 2004

 1
 2
 3
 4

 5
 6
 7
 8
 9
 10
 11

 12
 13
 14
 15
 16
 17
 18

 19
 20
 21
 22
 23
 24
 25

 26
 27
 28
 29
 30
 31

News Categories

▶ CPU

- **▶** Chipsets
- Memory
- ▶ Mainboards
- Video
- Storage
- Multimedia
- ▶ Mobile▶ Other
- <u>Otner</u>

Thursday, December 23, 2004

- ► [5:17 pm | Video] ATI'S Rialto Bridge: "Over the Next Couple of Months". ATI'S AGP-to-PCI Express Bridge to be Ready Soon
- ▶ [2:22 pm | CPU] IBM's, Apple's Computers to Get Virtualization in 2005. IBM to Incorporate Virtualization Technologies into Power Chips Next Year
- ▶ [10:39 am | other] Intel to Strengthen Concentration on Platform at Upcoming IDF. Intel Developer Forum Puts Tomorrow's Platforms Center Stage
- ▶ [7:55 am | Video] ATI and NVIDIA Proclaim Different Graphics Processors Architecture Goals. ATI Says

http://web.archive.org/web/20041224013927/http://www.xbitlabs.com/news/video/display/20041223075525.html [9/12/2015~3:49:31~PM]

ATI 2080 LG v. ATI IPR2015-00326 sport Shader Model 3.0 – pixel shaders 3.0 and vertex shaders 3.0 – bringing additional programming capabilities to ATI's future graphics processors as well as some other innovations.

ATI's R5xx architecture will not resemble that of the previous generation products and NVIDIA's GeForce 6 architecture known as NV4x, particularly ATI will implement efficient flow-control, a crucial feature for pixel shaders 3.0, that will not bring speed penalty it does on existing SM3.0 hardware, according to sources. The future of the graphics hardware lies in higher number of ALUs ops per texture ops, unified pixel and vertex shaders as well as some other requirements of Microsoft Windows Longhorn operating system, such as virtualisation and context switches. While ATI agrees on the long-term goals for its roadmap, it does not name feature-set of actual products and says all the architectural changes will be implemented gradually, not at once.

Some sources claim that the R500 is a code-name of ATI's graphics processor that will be submitted for Microsoft's next Xbox console. The shader core of the R500 was reported to have 48 Arithmetic Logic Units (ALUs) that can execute 64 simultaneous threads on groups of 64 vertices or pixels. ALUs are automatically and dynamically assigned to either pixel or vertex processing depending on load. The ALUs can each perform one vector and one scalar operation per clock cycle, for a total of 96 shader operations per clock cycle. Texture loads can be done in parallel to ALU operations. At peak performance, the GPU can issue 48 billion shader operations per second, it was said.

The R520 is also expected to feature advanced memory interface, presumably supporting <u>GDDR4 memory</u>.

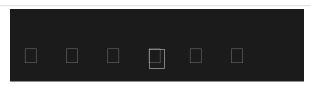
NVIDIA: Plans Unclear

While NVIDIA remains extremely tight-lipped over its future products, it is known that the company is readying its code-named NV47 visual processing unit, a massively revamped GeForce 6 architecture with 24 pixel pipelines. The NV47 is expected to be released sometime in Spring, 2005, but it is unknown whether NVIDIA is ahead, or behind ATI's R520 product. NVIDIA also reportedly plans to release a chip called NV48 in O2 2005.

The status of NVIDIA's future architecture code-named NV50 is also uncertain: some reported recently that the chip had been cancelled, but officials decline to confirm or deny the information.

Related news:

- ▶ <u>NVIDIA's PureVideo Accelerates Media on the PC.</u> NVIDIA Finally Delivers GeForce6 Video Engine
- ▶ <u>Gigabyte Plugs Two Graphics Processors on One Graphics Card.</u>
 Gigabyte 3D1 with Two GeForce 6600 GT Outperforms Top VPUs
- NVIDIA's NV48 to be Made at TSMC, Says Article. NVIDIA's NV48 May Still Be in Plans, Contrary to Reports
- ▶ <u>Industry Welcomes NVIDIA's New Low-End Graphics Cards.</u> NVIDIA's GeForce 6200 with TurboCache Set to Redefine Entry-Level
- ▶ <u>Samsung Tests Memory for 512MB Graphics Cards.</u> Samsung Samples 512Mb GDDR3 Chips
- ▶ NVIDIA Strikes PCI Express Graphics Cards Promotion Deal with Intel. Intel Corp. to Supply NVIDIA's Graphics Cards with its Mainboards
- ▶ NVIDIA Begins to Certify Hardware for Multi-GPU Deployments. NVIDIA Launches SLI Certification and Logo Program



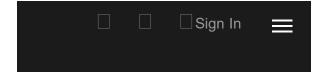
0002

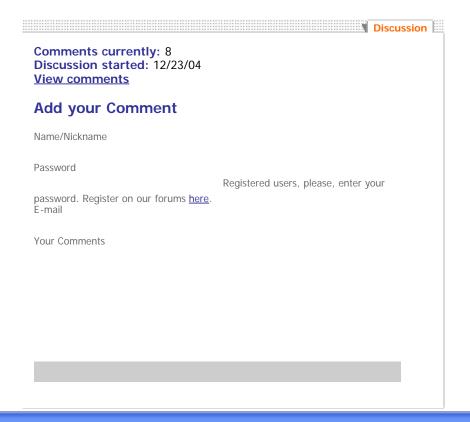
http://web.archive.org/web/20041224013927/http://www.xbitlabs.com/news/video/display/20041223075525.html[9/12/2015 3:49:31 PM]

Unified Rendering Engine – the Way to Go, NVIDIA Disagrees

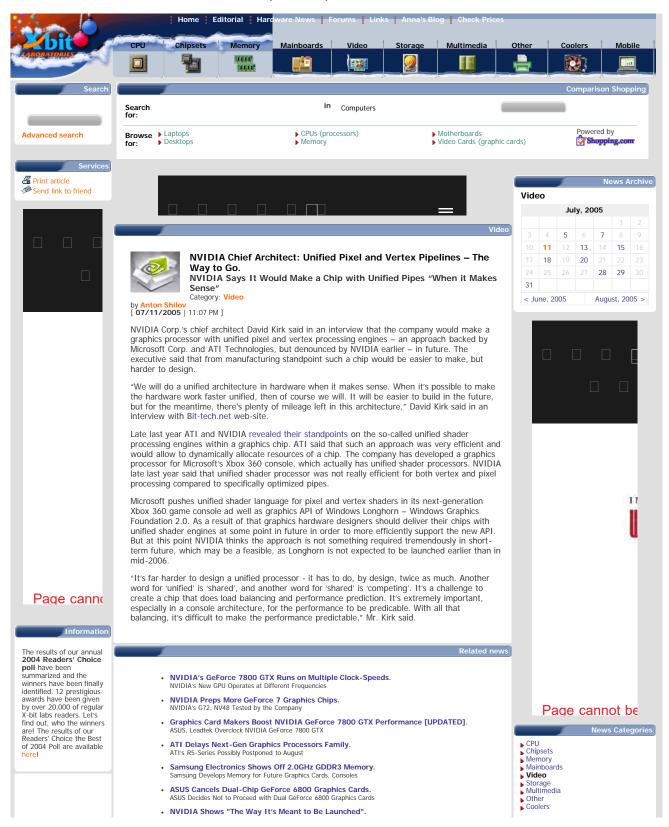
► [5:51 am | Chipsets] Intel to Remove DDR2 Support from Forthcoming Mainstream Chipsets. Intel's 915PL and 915GL Chipsets to Lack DDR2 Support

News Archive



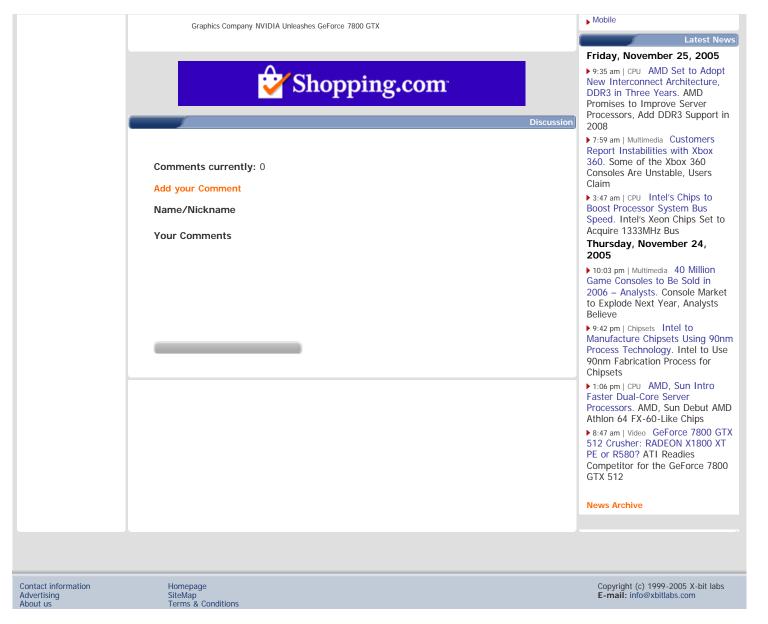


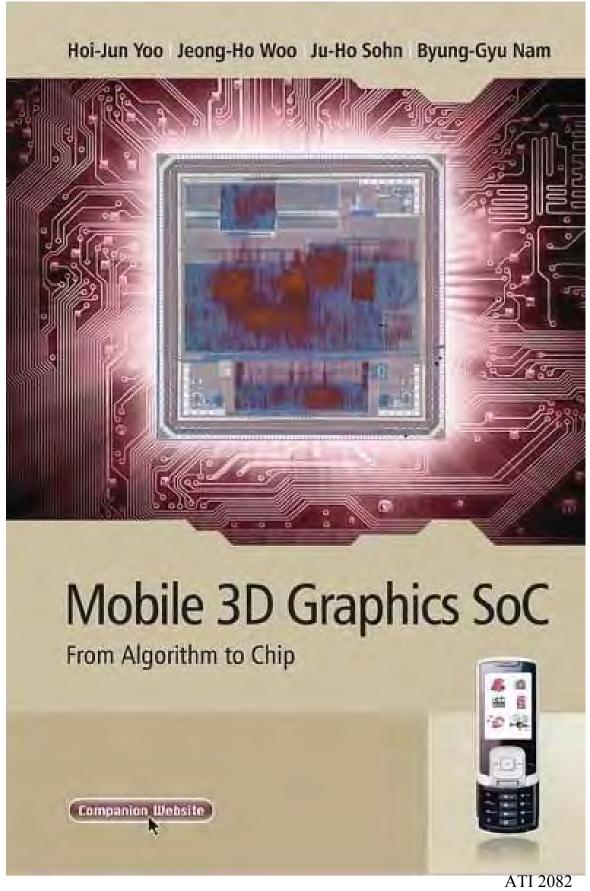
Copyright (c) 1999-2004 X-bit labs **E-mail:** info@xbitlabs.com Contact information Advertising About us Homepage SiteMap Terms & Conditions



http://web.archive.org/web/20051126060929/http://www.xbitlabs.com/news/video/display/20050711230734.html[9/12/2015 3:52:10 PM]

ATI 2081 LG v. ATI IPR2015-00326





Page 1 of 7

ATI 2082 LG v. ATI IPR2015-00326

MOBILE 3D GRAPHICS SoC

MOBILE 3D GRAPHICS SoC

From Algorithm to Chip

Jeong-Ho Woo

Korea Advanced Institute of Science and Technology, Republic of Korea

Ju-Ho Sohn

LG Electronics Institute of Technology, Republic of Korea

Byeong-Gyu Nam

Samsung Electronics, Republic of Korea

Hoi-Jun Yoo

Korea Advanced Institute of Science and Technology, Republic of Korea



John Wiley & Sons (Asia) Pte Ltd

Visit our Home Page on www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as expressly permitted by law, without either the prior written permission of the Publisher, or authorization through payment of the appropriate photocopy fee to the Copyright Clearance Center. Requests for permission should be addressed to the Publisher, John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop, #02-01, Singapore 129809, tel: 65-64632400, fax: 65-64646912, email: enquiry@wiley.com.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The Publisher is not associated with any product or vendor mentioned in this book. All trademarks referred to in the text of this publication are the property of their respective owners.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstrasse 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons Canada Ltd, 5353 Dundas Street West, Suite 400, Toronto, ONT, M9B 6H8, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Mobile 3D graphics SoC: from algorithm to chip / Jeong-Ho Woo ... [et al.].

p. cm.

Includes index.

ISBN 978-0-470-82377-4 (cloth)

- 1. Computer graphics. 2. Mobile computing. 3. Systems on a chip. 4. Three dimensional display systems.
- I. Woo, Jeong-Ho.

T385.M62193 2010

621.3815-dc22

2009049311

ISBN 978-0-470-82377-4 (HB)

Typeset in 10/12pt Times by Thomson Digital, Noida, India.

Printed and bound in Singapore by Markono Print Media Pte Ltd, Singapore.

This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Capacity	Size (μm ²)	AC current (mA) ^a	Peak current (mA)	Standby current (mA)
32 × 16 (512 b)	316 × 124	12.5	154.1	0.002
$64 \times 16 \ (1 \ \text{Kb})$	559×124	14.9	144.8	0.002
$128 \times 16 \ (2 \text{ Kb})$	559×134	14.9	144.8	0.002
$256 \times 16 \ (4 \text{ Kb})$	563×154	15.0	144.8	0.002
$512 \times 16 \ (1 \text{ KB})$	568×195	15.0	145.0	0.002
$1024 \times 16 \ (2 \text{ KB})$	753×125	13.0	183.4	0.003
$32 \times 32 \ (1 \ \text{Kb})$	559×124	20.8	271.5	0.003
$64 \times 32 \ (2 \text{ Kb})$	1044×124	25.5	277.0	0.003
$128 \times 32 \ (4 \ \text{Kb})$	1044×134	25.6	277.1	0.002
$256 \times 32 \ (1 \text{ KB})$	1044×154	25.6	277.2	0.002
$512 \times 32 \ (2 \text{ KB})$	1053×195	25.6	277.4	0.002
$1024 \times 32 \ (4 \text{ KB})$	1451×125	24.8	354.5	0.006

Table 5.4 SRAM characteristics in 0.13 μm CMOS processor (200 MHz, 1.2 V)

5.2.3 Mobile Unified Shader

As shaders have developed, vertex shaders and pixel shaders have similar instruction set architecture and register files, except for some unique instructions such as texture sampling. Therefore a unified shader architecture, which can compute vertex shading and pixel shading with the same hardware, has been developed to reduce

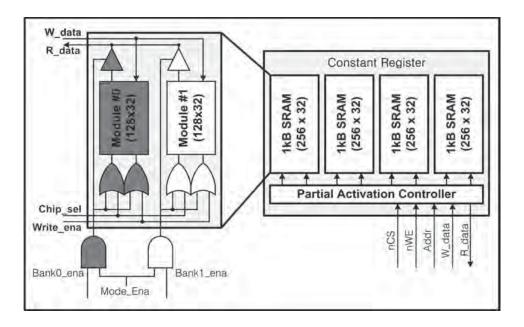


Figure 5.16 Partial activation

^a AC current is measured at 25% read, 25% write, 50% idle state.

design complexity and turnaround time. The first unified shader was implemented in Xenos by ATI for X-Box 360 [26]. In PC and console devices, a few tens of unified shaders are integrated and controlled with multi-thread control. Therefore, programmable graphics operations can be mapped to those unified shaders dynamically in real time, and the 3D graphics processor with unified shader can utilize the hardware resources more efficiently than conventional architecture with vertex shader and pixel shader.

In the mobile environment, a fully programmable 3D graphics pipeline is required. Owing to the need for low power consumption and small area, the conventional architecture with separate vertex shader and pixel shader is hard to implement. Since a unified shader can compute vertex shading and pixel shading in a single hardware, it is a good solution for programmable 3D graphics [21]. Figure 5.17a shows the block diagram of the 3D graphics processor, in which the unified shader performs vertex and pixel shading and other blocks perform other operations of the 3D graphics pipeline such as clipping, rasterization, and blending.

The unified shader is a 4-way SIMD processor. It consists of $128 \, \text{b}$, 4×32 -bit SIMD datapath, a special functional unit, a texture engine, a low-power lighting engine, register files, and control logic, as shown in Figure 5.17b. The SIMD datapath is responsible for vector and matrix arithmetic operations such as addition (ADD), multiplication (MUL), and inner product (DOT), and the special functional unit is dedicated to special functional scalar operations such as logarithm (LOG), exponent

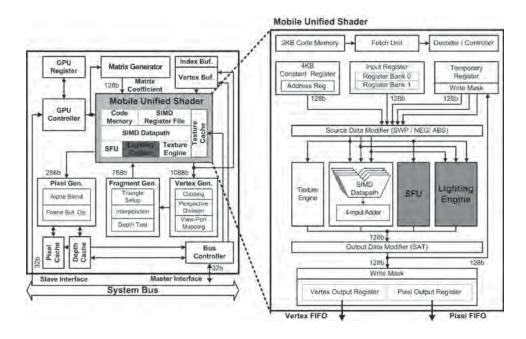


Figure 5.17 Mobile unified shader: (a) 3D graphics processor, and (b) mobile unified shader

(EXP), reciprocal (RCP), and reciprocal square-root (RSQ). The texture engine performs texture address generation, texture fetching, and texture filtering.

Since the single shader performs both vertex shading and pixel shading, task scheduling is crucial. Figure 5.18a shows a data flow diagram of the programmable 3D graphics pipeline. In conventional architecture, the primitive vertices are computed in the vertex shader for per-vertex operations such as transformation and lighting. After per-vertex operations, vertex generator and fragment generator perform clipping and

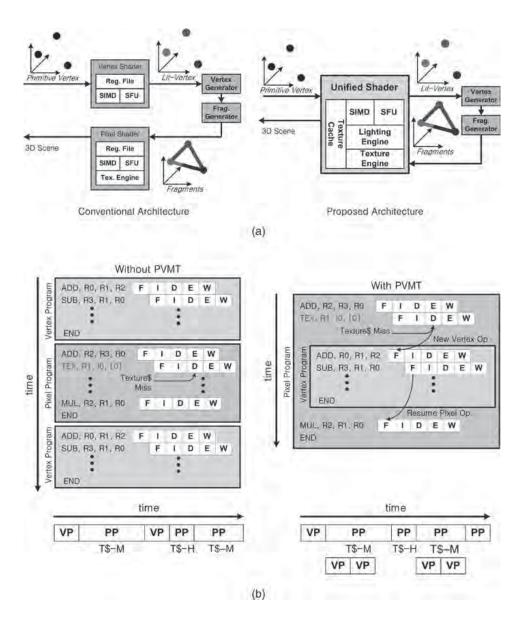
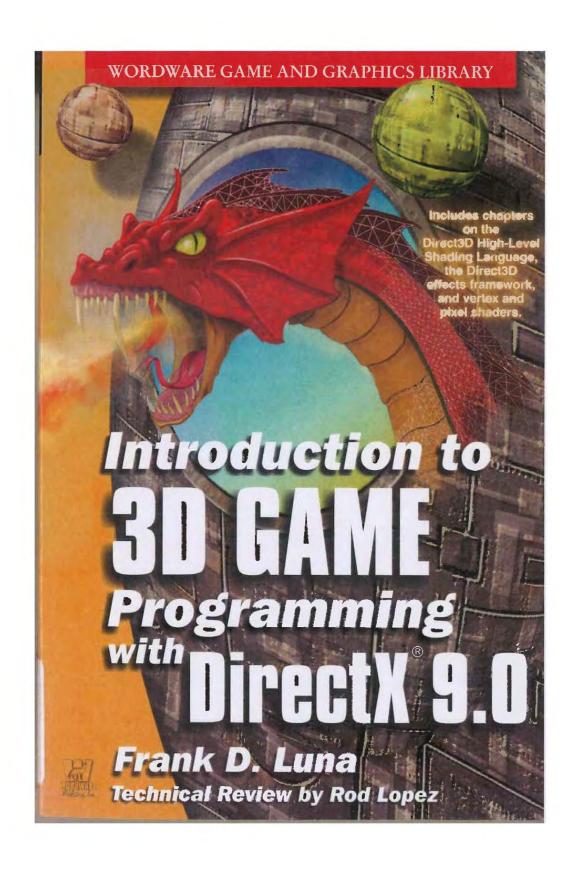


Figure 5.18 Pixel-vertex multi-threading: (a) data flow diagram, and (b) pixel-vertex multi-threading



ATI 2083 LG v. ATI IPR2015-00326

Introduction to 3D Game Programming with DirectX® 9.0

Frank D. Luna
Technical review by Rod Lopez

Wordware Publishing, Inc.

Library of Congress Cataloging-in-Publication Data

Luna, Frank D.

Introduction to 3D game programming with DirectX 9.0 / by Frank D. Luna.

p. cm.

ISBN-10: 1-55622-913-5 (pbk.)

1. Computer games--Programming. 2. DirectX. I. Title.

QA76.76.C672L83 2003

794.8'15268--dc21

2003005834

CIP

© 2003, Wordware Publishing, Inc. All Rights Reserved

> 2320 Los Rios Boulevard Plano, Texas 75074

No part of this book may be reproduced in any form or by any means without permission in writing from Wordware Publishing, Inc.

Printed in the United States of America

ISBN-13: 978-1-55622-913-8 ISBN-10: 1-55622-913-5

10 9 8 7 6 5 0305

DirectX is a registered trademark of Microsoft Corporation in the United States and/or other countries.

All brand names and product names mentioned in this book are trademarks or service marks of their respective companies. Any omission or misuse (of any kind) of service marks or trademarks should not be regarded as intent to infringe on the property of others. The publisher recognizes and respects all marks used by companies, manufacturers, and developers as a means to distinguish their products.

All inquiries for volume purchases of this book should be addressed to Wordware Publishing, Inc., at the above address. Telephone inquiries may be made by calling:

(972) 423-0090

```
const D3DXCOLOR YELLOW( D3DCOLOR_XRGB(255, 255, 0) );
const D3DXCOLOR CYAN( D3DCOLOR_XRGB(0, 255, 255) );
const D3DXCOLOR MAGENTA( D3DCOLOR_XRGB(255, 0, 255) );
```

4.2 Vertex Colors

The color of a primitive is determined by the color of the vertices that make it up. Therefore, we must add a color member to our vertex data structure. Note that a D3DCOLORVALUE type cannot be used here because Direct3D expects a 32-bit value to describe the color of a vertex. (Acually, by using a vertex shader we could use 4D color vectors for the vertex color, and thereby gain 128-bit color, but that is getting ahead of ourselves for now. Vertex shaders are covered in Chapter 17.)

```
struct ColorVertex
{
    float _x, _y, _z;
    D3DCOLOR _color;
    static const DWORD FVF;
}
const DWORD ColorVertex::FVF = D3DFVF_XYZ | D3DFVF_DIFFUSE;
```

4.3 Shading

Shading occurs during rasterization and specifies how the vertex colors are used to compute the pixel colors that make up the primitive. There are two shading modes that are presently used: flat shading and Gouraud shading.

With flat shading, the pixels of a primitive are uniformly colored by the color specified in the *first* vertex of the primitive. So the triangle formed by the following three vertices would be red, since the first vertex color is red. The colors of the second and third vertices are ignored with flat shading.

```
ColorVertex t[3];
t[0]._color = D3DCOLOR_XRGB(255, 0, 0);
t[1]._color = D3DCOLOR_XRGB(0, 255, 0);
t[2]._color = D3DCOLOR_XRGB(0, 0, 255);
```

Flat shading tends to make objects appear blocky because there is no smooth transition from one color to the next. A much better form of shading is called Gouraud shading (also called smooth shading). With Gouraud shading, the colors at each vertex are interpolated linearly across the face of the primitive. Figure 4.2 shows a red flat shaded triangle and a triangle colored using Gouraud shading.

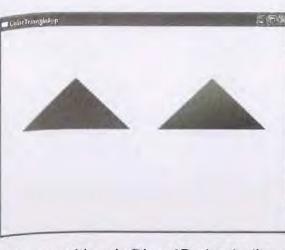


Figure 4.2: On the left is a triangle colored red with flat shading. On the right is a triangle with vertex colors red, green, and blue; notice that with Gouraud shading, the vertex colors are interpolated across the triangle.

Like many things in Direct3D, the shading mode is controlled through the Direct3D state machine.

```
// set flat shading
Device->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_FLAT);

// set Gouraud shading
Device->SetRenderState(D3DRS_SHADEMODE, D3DSHADE GOURAUD);
```

4.4 Sample Application: Colored Triangle

The sample program for this chapter demonstrates a triangle colored using flat shading and a triangle colored using Gouraud shading. It renders the image shown in Figure 4.2. First we add the following global variables:

```
DJDXMATRIX World;
|Direct3DVertexBuffer9* Triangle = 0;
```

We include a D3DXMATRIX that is used to store the world transformation of the triangles that we are going to draw. The Triangle variable is the vertex buffer that stores the vertex data of a triangle. Notice that we only have to store the geometry of one triangle because we can draw it multiple times at different positions in the world using the world matrix

The Setup method creates the vertex buffer and fills it with the data of a triangle with colored vertices. The first vertex in the triangle is full-intensity red (255), the second is full-intensity green (255), and the third is full-intensity blue (255). Finally, we disable lighting for this sample. Notice that this sample uses the new ColorVertex structure, as explained in section 4.2.

```
bool Setup()
      // create vertex buffer
      Device->CreateVertexBuffer(
           3 * sizeof(ColorVertex),
           D3DUSAGE WRITEONLY,
           ColorVertex::FVF,
           D3DPOOL MANAGED,
           &Triangle,
           0);
      // fill the buffers with the triangle data
     ColorVertex* v;
     Triangle->Lock(0, 0, (void**)&v, 0);
     v[0] = ColorVertex(-1.0f, 0.0f, 2.0f, D3DCOLOR_XRGB(255, 0, 0));
     v[1] = ColorVertex( 0.0f, 1.0f, 2.0f, D3DCOLOR_XRGB(0, 255, 0));
v[2] = ColorVertex( 1.0f, 0.0f, 2.0f, D3DCOLOR_XRGB(0, 0, 255));
     Triangle->Unlock();
     // set projection matrix
     D3DXMATRIX proj;
     D3DXMatrixPerspectiveFovLH(
           &proj,
           D3DX PI * 0.5f, // 90 - degree
           (float) Width / (float) Height,
           1.0f,
           1000.0f);
     Device->SetTransform(D3DTS PROJECTION, &proj);
     // set the render states
     Device->SetRenderState(D3DRS_LIGHTING, false);
     return true;
```

Then, the Display function draws Triangle twice in two different positions and with different shade modes. The position of each triangle is controlled with the world matrix—World.

```
Part II
```

```
Device->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_FLAT);
    Device->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 1);

// draw the triangle to the right with gourand shading
    D3DXMatrixTranslation(&World, 1.25f, 0.0f, 0.0f);
    Device->SetTransform(D3DTS_WORLD, &World);

Device->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_GOURAUD);
    Device->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 1);

Device->EndScene();
    Device->Present(0, 0, 0, 0);
}
return true;
```

// draw the triangle to the left with flat shading
p3DXMatrixTranslation(&World, -1.25f, 0.0f, 0.0f);

Device->SetTransform(D3DTS WORLD, &World);

4.5 Summary

- Colors are described by specifying an intensity of red, green, and blue. The additive mixing of these three colors at different intensities allows us to describe millions of colors. In Direct3D, we can use the D3DCOLOR, the D3DCOLORVALUE, or the D3DXCOLOR type to describe a color in code.
- We sometimes treat a color as a 4D vector (r, g, b, a). Color vectors are added, subtracted, and scaled just like regular vectors. On the other hand, dot and cross products do *not* make sense for color vectors, but component-wise multiplication does make sense for colors. The symbol \otimes denotes component-wise multiplication, and it is defined as: $(c_1, c_2, c_3, c_4) \otimes (k_1, k_2, k_3, k_4) = (c_1k_1, c_2k_2, c_3k_3, c_4k_4)$.
- We specify the color of each vertex, and then Direct3D uses the current shade mode to determine how these vertex colors are used to compute the pixel colors of the triangle during rasterization.
- With flat shading, the pixels of a primitive are uniformly colored by the color specified in the *first* vertex of the primitive. With Gouraud shading, the colors at each vertex are interpolated linearly across the face of the primitive.

Example:

Device->SetTexture (0, stonewall);



Note: In Direct3D, you can set up to eight textures that can be combined to create a more detailed image. This is called multitexturing. We do not use multitexturing in this book until Part IV; therefore we always set the texture's stage to 0, for now.

To disable a texture at a particular texturing stage, set pTexture to 0. For instance, if we don't want to render an object with a texture, we would write:

```
Device->SetTexture(0, 0);
renderObjectWithoutTexture();
```

If our scene has triangles that use different textures, we would have t_0 do something similar to the following code:

```
Device->SetTexture(0, _tex0);
drawTrisUsingTex0();
Device->SetTexture(0, _tex1);
drawTrisUsingTex1();
```

6.3 Filters

As mentioned previously, textures are mapped to triangles in screen space. Usually, the texture triangle is not the same size as the screen triangle. When the texture triangle is smaller than the screen triangle, the texture triangle is *magnified* to fit. When the texture triangle is larger than the screen triangle, the texture triangle is *minified* to fit. In both cases, distortion will occur. *Filtering* is a technique Direct3D uses to help smooth out these distortions.

Direct3D provides three different types of filters; each one provides a different level of quality. The better the quality, the slower it is, so you must make the trade-off between quality and speed. Texture filters are set with the IDirect3DDevice9::SetSamplerState method.

Nearest point sampling—This is the default filtering method and produces the worst-looking results, but it is also the fastest to compute. The following code sets nearest point sampling as the minification and magnification filter:

```
Device->SetSamplerState(0, D3DSAMP MAGFILTER, D3DTEXF POINT);
Device->SetSamplerState(0, D3DSAMP MINFILTER, D3DTEXF POINT);
```

■ Linear filtering —This type of filtering produces fairly good results and can be done very fast on today's hardware. It is

Once a light is registered, we can turn it on and off using what this next example illustrates:

```
Device->LightEnable(
        // the element in the light list to enable/disable
   true); // true = enable, false = disable
```

5.5 Sample Application: Lighting

The sample for this chapter creates the scene shown in Figure 5.7. It demonstrates how to specify vertex normals, how to create a material, and how to create and activate a directional light. Note that in this sample program we do not make use of the d3dUtility.h/cpp material and light functionality code because we want to show how it is done manually first. However, the rest of the samples in this book do use the material and light utility code.

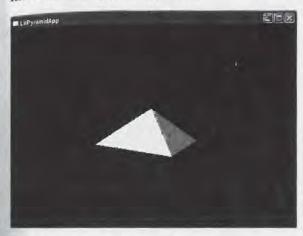


Figure 5.7: Screen shot from the LitPyramid sample

The steps for adding light to a scene are:

- 1. Enable lighting.
- 2. Create a material for each object and set the material before rendering the corresponding object.
- Create one or more light sources, set the light sources, and enable them.
- 4. Enable any additional lighting states, such as specular highlights.

First we instantiate a global vertex buffer that stores the pyramid's vertices:

IDirect3DVertexBuffer9* Pyramid = 0;

The Setup function contains all the code relevant to this chapter, so we omit the other functions to save space. It implements the previously discussed steps to add lighting to a scene. The Setup method starts by enabling lighting, which isn't necessary because it's enabled by default (but it doesn't hurt either).

```
bool Setup()
{
    Device->SetRenderState(D3DRS_LIGHTING, true);
```

Next, we create the vertex buffer, lock it, and specify the vertices that form triangles of the pyramid. The vertex normals were precomputed using the algorithm covered in section 5.3. Notice that while the triangles share vertices, they do not share normals; thus it is not very advantageous to use an index list for this object. For example, all the triangles share the peak point (0, 1, 0); however, for each triangle, the peak vertex normal points in a different direction.

```
Device->CreateVertexBuffer(
          12 * sizeof (Vertex),
          D3DUSAGE WRITEONLY,
          Vertex::FVF,
          D3DPOOL MANAGED,
          &Pyramid,
     // fill the vertex buffer with pyramid data
     Vertex* v;
     Pyramid->Lock(0, 0, (void**)&v, 0);
     // front face
     v[0] = Vertex(-1.0f, 0.0f, -1.0f, 0.0f, 0.707f, -0.707f);
     v[1] = Vertex( 0.0f, 1.0f, 0.0f, 0.0f, 0.707f, -0.707f);
     v[2] = Vertex(1.0f, 0.0f, -1.0f, 0.0f, 0.707f, -0.707f);
     // left face
     v[3] = Vertex(-1.0f, 0.0f, 1.0f, -0.707f, 0.707f, 0.0f);
     v[4] = Vertex(0.0f, 1.0f, 0.0f, -0.707f, 0.707f, 0.0f);
     v[5] = Vertex(-1.0f, 0.0f, -1.0f, -0.707f, 0.707f, 0.0f);
     // right face
     v[6] = Vertex(1.0f, 0.0f, -1.0f, 0.707f, 0.707f, 0.0f);
     v[7] = Vertex( 0.0f, 1.0f, 0.0f, 0.707f, 0.707f, 0.0f);
     v[8] = Vertex( 1.0f, 0.0f, 1.0f, 0.707f, 0.707f, 0.0f);
     // back face
     v[9] = Vertex( 1.0f, 0.0f, 1.0f, 0.0f, 0.707f, 0.707f);
     v[10] = Vertex( 0.0f, 1.0f, 0.0f, 0.0f, 0.707f, 0.707f);
     v[11] = Vertex(-1.0f, 0.0f, 1.0f, 0.0f, 0.707f, 0.707f);
     Pyramid->Unlock();
```

After we have generated the vertex data of our object, we describe how the object interacts with light by describing its materials. In this sample, the pyramid reflects white lights, emits no light, and produces some highlights.

```
D3DMATERIAL9 mtrl;
mtrl.Ambient = d3d::WHITE;
mtrl.Diffuse = d3d::WHITE;
mtrl.Specular = d3d::WHITE;
mtrl.Emissive = d3d::BLACK;
mtrl.Power = 5.0f;
Device->SetMaterial(&mtrl);
```

Second to last, we create and enable a directional light. The directional light rays run parallel to the x-axis in the positive direction. The light emits strong white diffuse light (dir.Diffuse = WHITE), weak white specular light (dir.Specular = WHITE * 0.3f), and a medium amount of white ambient light (dir.Ambient = WHITE * 0.6f).

```
p3pLIGHT9 dir;
::ZeroMemory(&dir, sizeof(dir));
dir.Type = D3pLIGHT_DIRECTIONAL;
dir.Diffuse = d3d::WHITE;
dir.Specular = d3d::WHITE * 0.3f;
dir.Ambient = d3d::WHITE * 0.6f;
dir.Direction = D3pXVECTOR3(1.0f, 0.0f, 0.0f);

Device->SetLight(0, &dir);
Device->LightEnable(0, true);
```

Finally, we set the state to renormalize normals and enable specular highlights.

```
Device->SetRenderState(D3DRS_NORMALIZENORMALS, true);
Device->SetRenderState(D3DRS_SPECULARENABLE, true);

// ... code to set up the view matrix and projection matrix
// omitted
return true;
```

5.6 Additional Samples

Three additional samples are included for this chapter in the companion files. They use the D3DXCreate* functions to create the 3D objects that compose the scene. The D3DXCreate* functions create vertex data with the format D3DFVF_XYZ | D3DFVF_NORMAL. In addition, these functions compute the vertex normals of each mesh for us. The additional samples demonstrate how to use directional lights, point lights, and spotlights. Figure 5.8 shows a screen shot from the directional light sample.

OpenGL - Lighting, Material, Shading and Texture Mapping

 $\begin{array}{c} \textbf{CS475 - Computer Graphics} \\ \textbf{Sumair Ahmed} \\ \textbf{IIT Bombay} \end{array}$

August 28, 2009

ATI 2084 LG v. ATI IPR2015-00326 This document briefs about the ways to deal with the prominent features existing in OpenGL. Here we discuss the Lighting, Shading, Material and Texture Mapping in programming perspective of OpenGL.

Lighting

Light makes the scene look real. Effectively, in a real world, light is present everywhere. Objects also have their own reaction in presence of a light. OpenGL supports a number of lighting effects - Directional, Spot, Ambient lights and attenuation. The Lighting and the Shading models define the light in OpenGL. But before setting these up, lighting mode has to be enabled. Lighting support needs the depth buffer to be enabled.

```
glEnable (GL\_DEPTH\_TEST);

glEnable (GL\_LIGHTING);
```

The above function glEnable can enable many features of OpenGL; the feature you want to enable is provided as an input parameter. The above code enables the lighting, however we need lights as well. OpenGL has direct support for about 8 lights . To enable a light, call:

```
glEnable(GL\_LIGHT0);
```

Likewise, LIGHT1 or 2..8 shall be enabled when you want to handle multiple lights. Similar to glEnable function, OpenGL also has glDisable function that disables the features set before by glEnable. These functions turn a particular feature ON or OFF. OpenGL also provides a set of specific global properties to specify the visual behavior of the lighting model. This is done in two ways:

```
glLightModelf(GLenum\ pname, GLfloat\ param); // scalar params glLightModelfv(GLenum\ pname, const\ GLfloat\ params); // vector params
```

For Instance, suppose you want to have a *GLOBAL* Ambient light that casts on all the rendered objects. The following code sets the *global_ambient* colour on all the objects.

```
Glfloat \ global\_ambient[] = \{0.5f, 0.5f, 0.5f, 1.0f\} \ // R,G,B,Alpha \ glLightModel fv (GL\_LIGHT\_MODEL\_AMBIENT, global\_ambient);
```

Similarly, we have GL_LIGHT_MODEL_TWO_SIDE and GL_LIGHT_MODEL_LOCAL_VIEWER parameters. The first parameter defines if the light is to be applied on two sides of an object or one side. Second defines how specular component is calculated. The specular highlight depends on the direction from the vertex to the viewpoint and also the direction from the vertex to the light source. So, the highlight depends on the eye position. With a default infinite viewpoint, the direction from the vertex to the viewpoint remains the same for all vertices. 0 is considered as infinite viewpoint, while 1 is local view point.

```
glLightModelf (GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
```

There are two types of light properties you should consider while programming the lighting model in OpenGL. The first type describes a light source and the second type describes the light reflected by the material of an object's surface. These four independent models define the light - $GL_AMBIENT$, $GL_DIFFUSE$, $GL_SPECULAR$ and $GL_EMISSIVE$

For every type of light source you need to call the glLightfv function with parameters. For example, To add a component of specular light to a light source, you would make the following function call:

```
GLfloat\ specular[] = \{1.0f, 1.0f, 1.0f, 1.0f, 1.0f\};

glLightfv\ (GL\_LIGHT0, GL\_SPECULAR, specular);
```

1

0002

The first three parameters of *specular* variable are the RGB values which can range between 0.0f and 1.0f. 0 being no colour, and 1 being full colour. The final parameter, Alpha, is used to define the translucency factor of an object and is used to simulate materials made out of translucent matter such as glass. The colour format here is often referred as RGBA format.

OpenGL also provides the light attenuation feature, that reduces the light intensity with the distance. Light Intensity should decrease with the distance to mimic the real world. It shall be inversely related to the distance. The Intensity attenuation formula would be :

```
Attenuated Intensity = Intensity of light Source * \frac{1}{C+L*d+Q*d^2} = Intensity of light Source * Attenuation Factor
```

where

```
d = distance between light source and the vertex C = GL\_Constant\_Attenuation (default is 1) L = GL\_Linear\_Attenuation (default is 0) Q = GL\_Quadratic\_Attenuation (default is 0)
```

are the three attenuation provided by OpenGL. In order to have an attenuation with the distance, you should set to a value different to 0 the linear or quadratic. It calculates an attenuation factor (between 0 and 1) which is multiplied to the ambient, diffuse and specular colours. By default, attenuation factor is 1, it means there is no attenuation w.r.t the distance.

```
glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.2f);
```

But assigning ambient, diffuse and specular types of light to a light source is not usually enough. You also have to specify the position of the light source. You can have Directional light source instead of the below Positional light source.

```
\begin{aligned} &Glfloat\ position1[] = \{-1.5f, 1.0f, -4.0f, 1.0f\};\ //x,y,z,w\\ &Glfloat\ position2[] = \{-1.05f, 1.0f, -1.0f, 0.0f\};\ //x,y,z,w\\ &glLightfv\left(GL\_LIGHT0, GL\_POSITION, position1\right);\\ &glLightfv\left(GL\_LIGHT0, GL\_POSITION, position2\right); \end{aligned}
```

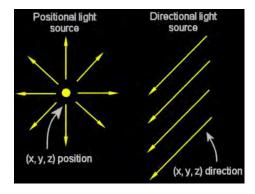


Figure 1: Positional and Directional light source

The light source can be a positional (w > 0) or directional (w = 0) light source depending on the w value. A positional light source is positioned at the location (x, y, z) as shown above. The source emits light from that particular location towards all directions. for example: lamp, bulb.

A directional one does not have any location. The source emits light from an infinite location, the rays are all parallel and have the direction (x, y, z). A directional light is not subject to attenuation since it is at an infinite distance. for example: Sun.

Shading

The Shading model is set up with a call to glShadeModel and can be either set to SMOOTH or FLAT model. The SMOOTH shading model specifies the use of Gouraud-shaded polygons to describe light while the FLAT shading model specifies the use of single-colored polygons.

```
glShadeModel(GL\_SMOOTH);
```

When SMOOTH model is selected, Lighting is evaluated at each vertex, and pixel colours are linearly interpolated across polygons. However, in FLAT model, Lighting is evaluated once for a polygon, and the resulting colour value is used for the complete object.

Material

This defines the reaction of an object when its surface is hit with the light. For example, some objects absorb a particular colour or reflects light. Usually when the lighting is enabled it is equally likely to assign material properties with the glMaterialf command as shown in the following code sample:

```
Glfloat\ mcolor[] = \{1.0f, 0.0f, 0.0f, 1.0f\};

glMaterialfv\ (GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE, mcolor);
```

The material property of an object (defined by RGB colour format), is usually the colour reflected by that object. The first parameter of the glMaterialfv command indicates which face of the polygon should reflect the light specified by mcolor. Apparently, there are two sides to a polygon - front and back. OpenGl provides two ways to specify a polygon in 3D space in order to decide the front face. The clockwise or counterclockwise direction describes which side is the front and which is the back. OpenGL lets you specify these rules with the glFrontFace command. The Following code denotes that counter clockwise direction direction of the polygons is considered to be Front Face.

```
glFrontFace(GL\_CCW);
```

glMaterial command should be called prior to defining the polygon's vertices to apply these material properties to the surface.

A convenient alternative to glmaterial is color tracking. Material properties are specified by merely calling the glColor command prior to each object or polygon. Also it has to be enabled.

```
\begin{split} glEnable\left(GL\_COLOR\_MATERIAL\right);\\ glColorMaterial\left(GL\_FRONT,GL\_AMBIENT\_AND\_DIFFUSE\right);\\ glColor3f(0.0f,0.0f,1.0f);//\text{ blue reflective properties} \end{split}
```

Example: This code would typically be placed with the OpenGL initialization code

```
//set the global lighting / shading glShadeModel\left(GL\_SMOOTH\right); // or GL\_FLAT glEnable\left(GL\_NORMALIZE\right); glEnable\left(GL\_LIGHTING\right); 
//set the global ambient light GLfloat\ ambient = \{.2, .2, .2, 1\}; glLightModelfv\left(GL\_LIGHT\_MODEL\_AMBIENT, globalAmb\right); 
//set up a light and enable it GLfloat\ diffuse[] = \{1, 0, 0, 1\}; GLfloat\ ambient[] = \{.5, 0, 0, 1\}; GLfloat\ specular[] = \{1, 1, 1, 1\};
```

```
\begin{split} &glLightfv\left(GL\_LIGHT0,GL\_DIFFUSE,diffuse\right);\\ &glLightfv\left(GL\_LIGHT0,GL\_AMBIENT,ambient\right);\\ &glLightfv\left(GL\_LIGHT0,GL\_SPECULAR,specular\right);\\ &glEnable\left(GL\_LIGHT0\right);\ //enable\ the\ light\\ &\ //set\ light\ position\\ &\ //set\ last\ term\ to\ 0\ for\ a\ spotlight\\ &Glfloat\ lightpos[] = \{1,1,1,1\};\\ &glLightfv\left(GL\_LIGHT0,GL\_POSITION,lightpos\right);\\ &\ //This\ code\ sets\ a\ simple\ material\ property\\ &GLfloat\ ambient[] = \{.5,0,0,1\};\\ &GLfloat\ specular[] = \{1,1,1,1\};\\ &\ //set\ params\ for\ front\ and\ back\ separately\ (GL\_BACK,GL\_FRONT\_AND\_BACK)\\ &glMaterialfv\left(GL\_FRONT,GL\_AMBIENT\_AND\_DIFFUSE,ambient\right);\\ &glMaterialfv\left(GL\_FRONT,GL\_SPECULAR,ambient\right); \end{split}
```

Example2: A Cube in an environment with two diffuse lights and an ambient light. Two Diffuse lights are of different colours (blue & green) whereas the ambient light is red colour.

```
\begin{split} GLfloatDiffuseLight1[] &= \{0,0,1\};\\ GLfloatDiffuseLight2[] &= \{0,1,0\};\\ GLfloatAmbientLight[] &= \{1,0,0\};\\ gLightfv\left(GL\_LIGHT0,GL\_DIFFUSE,DiffuseLight1\right);\\ glLightfv\left(GL\_LIGHT1,GL\_AMBIENT,AmbientLight\right);\\ glLightfv\left(GL\_LIGHT2,GL\_DIFFUSE,DiffuseLight2\right);\\ GLfloatLightPosition1[] &= \{0,0,3,0\};\\ GLfloatLightPosition2[] &= \{3,0,0,0\};\\ glLightfv\left(GL\_LIGHT0,GL\_POSITION,LightPosition1\right);\\ glLightfv\left(GL\_LIGHT2,GL\_POSITION,LightPosition2\right);\\ glLlookAt\left(3.0,-2.0,4.0,0.0,0.0,0.0,1.0,0.0\right); \end{split}
```

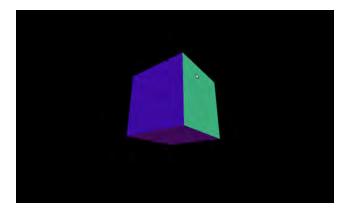


Figure 2: Output of Example2

You can see three colours on three sides of a cube. There is one diffuse light at position (3,0,0) and the other at (0,0,3). There is also an Ambient light of red colour. The eye position is at (3,-2,4) and is looking at (0,0,0) i.e., the center of cube.

Texture Mapping

Another feature in OpenGL is Texture Mapping feature where you can apply the textures to your geometry. Once a texture is uploaded to the video memory it can be used throughout the program. There are certain steps to be followed before a texture is readily available to the program. We first need a texture name. This is essentially a number that OpenGL uses to index all the different textures.

```
GLuint texture; // allocate a texture name qlGenTextures (1, &texture); //get a free texture id
```

Now that we have our texture name, it has to be bound before doing anything to it. Note that there are two forms of textures in OpenGL, 1D and 2D. You can load different textures, however only one is selected at a time.

```
// select our current texture glBindTexture (GL_TEXTURE_2D, texture);
```

Now we need to set some texture parameters and load the texture data on the current texture. OpenGL has four texture parameters to setup. Here, it defines several effects like bilinear, trilinear texture filtering, and mipmapping. We also can define whether the texture wraps over at the edges or is clamped at the ends.

```
// the texture wraps over at the edges (repeat)
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

// when texture area is large, bilinear filter the original
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

The default state of MIN_FILTER is $GL_LINEAR_MIPMAP_NEAREST$, if it is not defined. In such a case, the texture is considered incomplete and it renders a white texture on the object.

Also we need to set environment variables for the current texture. This tells the OpenGL how the texture should act when it is rendered into a scene.

```
glTexEnvf(GL\_TEXTURE\_ENV,GL\_TEXTURE\_ENV\_MODE,GL\_MODULATE);
```

Here it sets the active texture to $GL_MODULATE$. This attribute allows to apply effects such as lighting and colouring to your texture. If you would like to display the texture unchanged then replace it with GL_DECAL .

After all these parameters are set, OpenGL calls glTexImage2D that will upload the texture to the video memory and will be ready for us to use in our programs.

 $glTexImage2D(GL.TEXTURE_2D, level, internalFormat, width, height, border, format, type, ptexels);$

- internalFormat This tells OpenGL how many colour components are needed to represent internally from the texture that is uploaded.ex: GL_RGB
- format Format of the pixel data that will be uploaded. ex: GL_RGB
- $\bullet~type$ Type of data that will be uploaded. ex: $GL_UNSIGNED_BYTE$
- ptexels Pointer to the image data.

Note that after your call to glTexImage2D you can free this memory with free function since the texture is already uploaded into video memory. A good alternative to glTexImage2D is to build your texture mipmaps. This can be done by:

$gluBuild2DMipmaps(GL_TEXTURE_2D, 3, width, height, GL_RGB, \\ GL_UNSIGNED_BYTE, data);$

Now the texture is ready to be applied to your geometry, with all the above parameters set. Remember Texturing has to be enabled.

Example: Texture Quad

```
// enable texturing glEnable\ (GL\_TEXTURE\_2D); glBegin\ (GL\_QUADS); glTexCoord2d\ (0.0,0.0); glVertex2d\ (0.0,0.0); glVertex2d\ (1.0,0.0); glVertex2d\ (1.0,0.0); glTexCoord2d\ (1.0,1.0); glVertex2d\ (1.0,1.0); glTexCoord2d\ (0.0,1.0); glVertex2d\ (0.0,1.0); glVertex2d\ (0.0,1.0); glEnd\ ();
```

References

- $\bullet \ \ {\it OpenGL Tutorials-http://www.swiftless.com/tutorials/opengl/opengltuts.html}$
- \bullet Texture Mapping $http://www.opengl.org/wiki/Texture_Mapping$
- \bullet OpenGL Lighting http://www.falloutsoftware.com/tutorials/gl/gl8.htm
- $\bullet \ \ {\rm OpenGL\ Texture\ Mapping}\ \hbox{-}\ http://www.gamedev.net/reference/articles/article947.asp$
- $\bullet \ \ \text{OpenGL Texture Tutorial} \ \ \textit{http:} \ //www.null terminator.net/gltexture.html$

Microsoft*



Microsoft[®]

Computer Dictionary

Fifth Edition

- Fully updated with the latest technologies, terms, and acronyms
- · Easy to read, expertly illustrated
- Definitive coverage of hardware, software, the Internet, and more!



ATI 2085 LG v. ATI IPR2015-00326 PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 2002 by Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Cataloging-in-Publication Data Microsoft Computer Dictionary.--5th ed. p. cm. ISBN 0-7356-1495-4

1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002 004'.03--dc21

200219714

Printed and bound in the United States of America.

23456789 QWT 765432

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to mspinput@microsoft.com.

Active Desktop, Active Directory, ActiveMovie, ActiveStore, ActiveSync, ActiveX, Authenticode, BackOffice, BizTalk, ClearType, Direct3D, DirectAnimation, DirectDraw, DirectInput, DirectMusic, DirectPlay, DirectShow, DirectSound, DirectX, Entourage, FoxPro, FrontPage, Hotmail, IntelliEye, IntelliMouse, IntelliSense, JScript, MapPoint, Microsoft, Microsoft Press, Mobile Explorer, MS-DOS, MSN, Music Central, NetMeeting, Outlook, PhotoDraw, PowerPoint, SharePoint, UltimateTV, Visio, Visual Basic, Visual C++, Visual FoxPro, Visual InterDev, Visual J++, Visual SourceSafe, Visual Studio, Win32, Win32s, Windows, Windows Media, Windows NT, Xbox are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Acquisitions Editor: Alex Blanton Project Editor: Sandra Haynes

Body Part No. X08-41929

archive

application processor *n*. A processor dedicated to a single application.

application program n. See application.

application program Interface. *n. See* application programming interface.

application programming Interface *n*. A set of routines used by an application program to direct the performance of procedures by the computer's operating system. *Acronym:* API. *Also called:* application program interface.

application server *n*. **1.** A server program on a computer in a distributed network that handles the business logic between users and backend business applications or databases. Application servers also can provide transaction management, failover, and load balancing. An application server is often viewed as part of a three-tier application consisting of a front-end GUI server such as an HTTP server (first tier), an application server (middle tier), and a backend database and transaction server (third tier). *Also called:* appserver. *Compare* HTTP server (definition 1). **2.** Any machine on which an application-server program is running. *Also called:* appserver.

application service provider *n*. A third-party company or organization that hosts applications or services for individuals or business customers. The customer connects to a data center maintained by the application service provider (ASP) through Internet or private lines to access applications that would otherwise need to be housed on the customer's local servers or individual PCs. This arrangement allows the customer to free up disk space that would otherwise be taken by applications, as well as to access the most recent software updates. ASPs deliver solutions ranging from high-end applications to services for small and medium-sized businesses. *Acronym:* ASP.

application shortcut key *n*. A key or combination of keys that when pressed will quickly perform an action within an application that would normally require several user actions, such as menu selections. *Also called:* keyboard shortcut.

application software n. See application.

application-specific integrated circuit n_e See gate array. application suite n_e See suite (definition 1).

appserver n. See application server.

Aqua n. The graphical user interface (GUI) of Macintosh OS X. Aqua was designed to maintain familiarity and a comfort level for users of the earlier Macintosh system while allowing access to newer Macintosh OS X capabilities. The Aqua GUI features updated versions of Macintosh staples such as the Finder alongside new features like the Dock, a new type of organizational tool. See also Dock, Macintosh OS X.

arbitration *n*. A set of rules for resolving competing demands for a machine resource by multiple users or processes, *See also* contention.

.arc n. The file extension that identifies compressed archive files encoded using the Advanced RISC Computing Specification (ARC) format. See also compressed file.

arcade game *n*. **1.** A coin-operated computer game for one or more players that features high-quality screen graphics, sound, and rapid action. **2.** Any computer game designed to mimic the style of a coin-operated arcade game, such as games marketed for the home computer. *See also* computer game.

Archle *n*. An Internet utility for finding files in public archives obtainable by anonymous FTP. The master Archie server at McGill University in Montreal downloads FTP indexes from participating FTP servers, merges them into a master list, and sends updated copies of the master list to other Archie servers each day. Archie is a shortened form of *archive*. *See also* anonymous FTP, FTP¹ (definition 1). *Compare* Jughead, Veronica.

Archie client n. See Archie.

Archie server n. On the Internet, a server that contains Archie indexes to the names and addresses of files in public FTP archives. *See also* Archie, FTP¹ (definition 1), server (definition 2).

architecture *n*. **1.** The physical construction or design of a computer system and its components. *See also* cache, CISC, closed architecture, network architecture, open architecture, pipelining, RISC. **2.** The data-handling capacity of a microprocessor. **3.** The design of application software incorporating protocols and the means for expansion and interfacing with other programs.

archive¹ n. 1. A tape or disk containing files copied from another storage device and used as backup storage. 2. A compressed file. 3. A file directory on the Internet that is

CD. A permutation is a grouping of elements taken from a larger set with regard to the order of the elements. For example, in making permutations of two objects from the same set of four objects, there would be four candidates to choose from for the first selection (A), and three left over to choose from for the second selection (B), or 12 permutations in all: AB, AC, AD, BA, BC, BD, CA, CB, CD, DA, DB, DC. *See also* combinatorial explosion.

COM callable wrapper *n*. A proxy object generated by the runtime so that existing COM applications can use managed classes, including .NET Framework classes, transparently. *Acronym:* CCW.

COMDEX *n*. Any of a series of annual computer trade shows operated by Softbank COMDEX, Inc. One of these shows takes place in Las Vegas each November and is the largest computer trade show in the United States.

Comité Consultatif International Télégraphique et Téléphonique n. See CCITT.

comma-delimited file *n*. A data file consisting of fields and records, stored as text, in which the fields are separated from each other by commas. Use of comma-delimited files allows communication between database systems that use different formats. If the data in a field contains a comma, the field is further surrounded with quotation marks.

command *n*. An instruction to a computer program that, when issued by the user, causes an action to be carried out. Commands are usually either typed at the keyboard or chosen from a menu.

command buffer *n*. An area in memory in which commands entered by the user are kept. A command buffer can enable the user to repeat commands without retyping them completely, edit past commands to change some argument or correct a mistake, undo commands, or obtain a list of past commands. *See also* history, template (definition 4).

command button *n*. A control shaped like a pushbutton in a dialog box in a graphical user interface. By clicking a command button, the user causes the computer to perform some action, such as opening a file that has just been selected using the other controls in the dialog box.

COMMAND.COM *n*. The command interpreter for MS-DOS. *See also* command interpreter.

command-driven *adj.* Accepting commands in the form of code words or letters, which the user must learn. *Compare* menu-driven.

command-driven system *n*. A system in which the user initiates operations by a command entered from the console. *Compare* graphical user interface.

command Interpreter *n*. A program, usually part of the operating system, that accepts typed commands from the keyboard and performs tasks as directed. The command interpreter is responsible for loading applications and directing the flow of information between applications. In OS/2 and MS-DOS, the command interpreter also handles simple functions, such as moving and copying files and displaying disk directory information. *See also* shell¹.

Command key *n*. On the original Macintosh keyboard, a key labeled with the special symbol, sometimes called the propeller or puppy foot. This key is found on one or both sides of the Spacebar, depending on the version of the Apple keyboard. The key serves some of the same functions as the Control key on IBM keyboards. *See also* Control key.

command language *n*. The set of keywords and expressions that are accepted as valid by the command interpreter. *See also* command interpreter.

command line *n*. A string of text written in the command language and passed to the command interpreter for execution, *See also* command language,

command-line Interface *n*. A form of interface between the operating system and the user in which the user types commands, using a special command language. Although systems with command-line interfaces are usually considered more difficult to learn and use than those with graphical interfaces, command-based systems are usually programmable; this gives them flexibility unavailable in graphics-based systems that do not have a programming interface. *Compare* graphical user interface.

command mode *n*. A mode of operation in which a program waits for a command to be issued, *Compare* edit mode, insert mode.

command processing *n. See* command-driven system. **command processor** *n. See* command interpreter.

command prompt window *n*. A window displayed on the desktop used to interface with the MS-DOS operating

C

insider attack instruction set

Insider attack *n*. An attack on a network or system carried out by an individual associated with the hacked system. Insider attacks are typically the work of current or former employees of a company or organization who have knowledge of passwords and network vulnerabilities. *Compare* intruder attack.

Ins key n. See Insert key.

Install *vb.* To set in place and prepare for operation. Operating systems and application programs commonly include a disk-based installation, or setup, program that does most of the work of preparing the program to work with the computer, printer, and other devices. Often such a program can check for devices attached to the system, request the user to choose from sets of options, create a place for the program on the hard disk, and modify system startup files as necessary.

Installable device driver *n*. A device driver that can be embedded within an operating system, usually in order to override an existing, less-functional service.

Installable File System Manager *n*. In Windows 9x and Windows 2000, the part of the file system architecture responsible for arbitrating access to the different file system components. *Acronym:* IFS.

Installation program *n*. A program whose function is to install another program, either on a storage medium or in memory. An installation program, also called a setup program, might be used to guide a user through the often complex process of setting up an application for a particular combination of machine, printer, and monitor.

Installer n. A program, provided with the Apple Macintosh operating system, that allows the user to install system upgrades and make bootable (system) disks.

Instance *n*. An object, in object-oriented programming, in relation to the class to which it belongs. For example, an object *myList* that belongs to a class *List* is an instance of the class *List*. *See also* class, instance variable, instantiate, object (definition 2).

Instance variable *n*. A variable associated with an instance of a class (an object). If a class defines a certain variable, each instance of the class has its own copy of that variable. *See also* class, instance, object (definition 2), object-oriented programming.

Instantlate *vb.* To create an instance of a class. *See also* class, instance, object (definition 2).

Instant messaging *n*. A service that alerts users when friends or colleagues are on line and allows them to communicate with each other in real time through private online chat areas. With instant messaging, a user creates a list of other users with whom he or she wishes to communicate; when a user from his or her list is on line, the service alerts the user and enables immediate contact with the other user. While instant messaging has primarily been a proprietary service offered by Internet service providers such as AOL and MSN, businesses are starting to employ instant messaging to increase employee efficiency and make expertise more readily available to employees.

Institute of Electrical and Electronics Engineers n. See IEEE.

Instruction n. An action statement in any computer language, most often in machine or assembly language. Most programs consist of two types of statements: declarations and instructions. *See also* declaration, statement.

Instruction code n. See operation code.

Instruction counter n. See instruction register.

Instruction cycle *n*. The cycle in which a processor retrieves an instruction from memory, decodes it, and carries it out. The time required for an instruction cycle is the sum of the instruction (fetch) time and the execution (translate and execute) time and is measured by the number of clock ticks (pulses of a processor's internal timer) consumed.

Instruction mix *n*. The assortment of types of instructions contained in a program, such as assignment instructions, mathematical instructions (floating-point or integer), control instructions, and indexing instructions. Knowledge of instruction mixes is important to designers of CPUs because it tells them which instructions should be shortened to yield the greatest speed, and to designers of benchmarks because it enables them to make the benchmarks relevant to real tasks.

Instruction pointer n. See program counter.

Instruction register *n*. A register in a central processing unit that holds the address of the next instruction to be executed.

Instruction set n. The set of machine instructions that a processor recognizes and can execute. *See also* assembler, microcode.

frequently requested data) on computers running at bus speeds of 75 MHz or higher. Acronym. PB SRAM, See also burst (definition 2), L2 cache, pipelining, static RAM. Compare asynchronous static RAM, dynamic RAM, synchronous burst static RAM.

pipeline processing n. A method of processing on a computer that allows fast parallel processing of data. This is accomplished by overlapping operations using a pipe, or a portion of memory that passes information from one process to another. See also parallel processing, pipe (definition 1), pipelining (definition 3).

pipelining n. 1. A method of fetching and decoding instructions (preprocessing) in which, at any given time, several program instructions are in various stages of being fetched or decoded. Ideally, pipelining speeds execution time by ensuring that the microprocessor does not have to wait for instructions; when it completes execution of one instruction, the next is ready and waiting. See also superpipelining. 2. In parallel processing, a method in which instructions are passed from one processing unit to another, as on an assembly line, and each unit is specialized for performing a particular type of operation. 3. The use of pipes in passing the output of one task as input to another until a desired sequence of tasks has been carried out. See also pipe (definition 1), pour.

piracy n. 1. The theft of a computer design or program. 2. Unauthorized distribution and use of a computer program.

.pit n. A file extension for an archive file compressed with PackIT. See also PackIT.

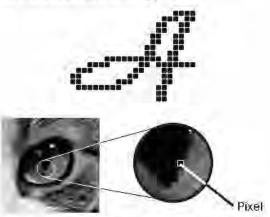
pitch n A measure, generally used with monospace fonts, that describes the number of characters that fit in a horizontal inch. See also characters per inch, screen pitch. Compare point¹ (definition 1).

PivotChart adj. A graphical tool in Microsoft Excel or Access that can be used to display data from a list or database in chart form. Based on user-selected information incorporated in an Excel PivotTable report or list, a PivotChart report provides the ability to chart the data interactively—for example, to "pivot" the chart's point of view from product sales by category to product sales by region or by salesperson. See also PivotTable.

PivotTable adj. An interactive table in Microsoft Excel or Access that can show the same data from a list or a database in more than one arrangement. A user can manipulate the rows and columns in a PivotTable to view or summarize the information in different ways for purposes of analysis. In Excel, a PivotTable report is the basis for creating a PivotChart report that displays the same data in chart form. See also PivotChart.

pivot year n. In Year 2000 windowing, a date in a 100-year period that serves as the point from which correct dates can be calculated in systems or software that can store only 2-digit years. For example, a pivot year of 1970 means that the numbers 70 through 99 are interpreted as the years 1970 to 1999, and the numbers 00 through 69 as the years 2000 through 2069. See also windowing.

pixel n. Short for picture (**pix**) element. One spot in a rectilinear grid of thousands of such spots that are individually "painted" to form an image produced on the screen by a computer or on paper by a printer. A pixel is the smallest element that display or print hardware and software can manipulate in creating letters, numbers, or graphics. See the illustration. Also called pel.



Pixel. The letter A is actually made up of a pattern of pixels in a grid, as is the cat's eye.

pixel image n The representation of a color graphic in a computer's memory. A pixel image is similar to a bit image, which also describes a screen graphic, but a pixel image has an added dimension, sometimes called depth, that describes the number of bits in memory assigned to each on-screen pixel.

406

point of sale Pong

point of sale n. See POS.

point-to-point configuration n. A communications link in which dedicated links exist between individual origins and destinations, as opposed to a point-to-multipoint configuration, in which the same signal goes to many destinations (such as a cable TV system), or a switched configuration, in which the signal moves from the origin to a switch that routes the signal to one of several possible destinations. Also called: point-to-point connection.

point-to-point connection n. See point-to-point configuration

point-to-point message system *n*. In Sun Microsystems's J2EE network platform, a messaging system that uses message queues to store asynchronous, formatted data for coordinating enterprise applications. Each message is addressed to a specific queue, and client applications retrieve messages from the queues. *See also* asynchronous, J2EE.

Point-to-Point Protocol n. See PPP.

point-to-point tunneling n. A means of setting up secure communications over an open, public network such as the Internet. *See also* PPTP.

Point-to-Point Tunneling Protocol n. See PPTP.

Polsson distribution *n*. A mathematical curve often used in statistics and simulation to represent the likelihood of some event occurring, such as the arrival of a customer in a queue, when the average likelihood is known. This distribution, named after the French mathematician S. D. Poisson, is simpler to calculate than the normal and binomial distributions. *See also* binomial distribution, normal distribution.

poke *vb.* To store a byte into an absolute memory location. PEEK (read a byte from memory) and POKE commands are often found in programming languages, such as Basic, that do not normally allow access to specific memory locations.

polar coordinates n. Coordinates of the form (r, q) used to locate a point in two dimensions (on a plane). The polar coordinate r is the length of the line that starts at the origin and ends at the point, and q (Greek theta) is the angle between that line and the positive x-axis. Compare Cartesian coordinates.

polarity *n*. The sign of the potential (voltage) difference between two points in a circuit. When a potential difference exists between two points, one point has a positive polarity and the other a negative polarity. Electrons flow from negative to positive; by convention, however, current is considered to flow from positive to negative.

polarized component *n*. A circuit component that must be installed with its leads in a particular orientation with respect to the polarity of the circuit. Diodes, rectifiers, and some capacitors are examples of polarized components.

polarizing filter *n*. A transparent piece of glass or plastic that polarizes the light passing through it; that is, it allows only waves vibrating in a certain direction to pass through. Polarizing filters are often used to reduce glare on monitor screens. *See also* glare filter.

Pollsh notation *n. See* prefix notation.

polling n. See autopolling.

polling cycle *n*. The time and sequence required for a program to poll each of its devices or network nodes. *See also* autopolling.

polygon *n*. Any two-dimensional closed shape composed of three or more line segments, such as a hexagon, an octagon, or a triangle. Computer users encounter polygons in graphics programs.

polyline *n*. An open shape consisting of multiple connected segments. Polylines are used in CAD and other graphics programs. *See also* CAD.

polymorphism *n*. In an object-oriented programming language, the ability to redefine a routine in a derived class (a class that inherited its data structures and routines from another class). Polymorphism allows the programmer to define a base class that includes routines that perform standard operations on groups of related objects, without regard to the exact type of each object. The programmer then redefines the routines in the derived class for each type, taking into account the characteristics of the object. *See also* class, derived class, object (definition 2), object-oriented programming.

Pong *n*. The first commercial video game, a table tennis simulation, created by Nolan Bushnell of Atari in 1972.



preventive maintenance printed circuit board.

preventive maintenance n. Routine servicing of hardware intended to keep equipment in good operating condition and to find and correct problems before they develop into severe malfunctions.

preview *n*. In word processors and other applications, the feature that formats a document for printing but displays it on the video monitor rather than sending it directly to the printer.

PRI *n*. Acronym for Primary Rate Interface. One of two ISDN transmission rate services (the other is the basic rate interface, BRI). PRI has two variations. The first, which operates at 1.536 Mbps, transmits data over 23 B channels and sends signaling information at 64 Kbps over one D channel in the United States, Canada, and Japan. The second, which operates at 1.984 Mbps, transmits data over 30 B channels and sends signaling information at 64 Kbps over one D channel in Europe and Australia. *See also* BRI, ISDN.

primary channel n. The data-transmission channel in a communications device, such as a modem. *Compare* secondary channel.

Primary Domain Controller n. 1. In Windows NT, a database providing a centralized administration site for resources and user accounts. The database allows users to log onto the domain, rather than onto a specific host machine. A separate account database keeps track of the machines in the domain and allocates the domain's resources to users. 2. In any local area network, the server that maintains the master copy of the domain's user accounts database and that validates logon requests, Acronym: PDC.

primary key n. In databases, the key field that serves as the unique identifier of a specific tuple (row) in a relation (database table). *Also called:* major key. *See also* alternate key (definition 1), candidate key. *Compare* secondary key.

Primary Rate Interface n. See PRI.

primary storage *n*. Random access memory (RAM); the main general-purpose storage region to which the microprocessor has direct access. A computer's other storage options, such as disks and tape, are called *secondary storage* or (sometimes) *backing storage*.

primitive *n*. **1.** In computer graphics, a shape, such as a line, circle, curve, or polygon, that can be drawn, stored, and manipulated as a discrete entity by a graphics program. A primitive is one of the elements from which a large graphic design is created. **2.** In programming, a fundamen-

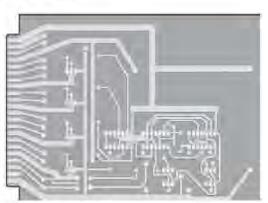
tal element in a language that can be used to create larger procedures that do the work a programmer wants to do.

print *vb*. In computing, to send information to a printer. The word is also sometimes used in the sense of "show me" or "copy this." For example, the PRINT statement in Basic causes output to be displayed (printed) on the screen. Similarly, an application program that can be told to print a file to disk interprets the command as an instruction to route output to a disk file instead of to a printer.

print buffer *n*. A section of memory to which print output can be sent for temporary storage until the printer is ready to handle it. A print buffer can exist in a computer's random access memory (RAM), in the printer, in a separate unit between the computer and the printer, or on disk. Regardless of its location, the function of a print buffer is to free the computer for other tasks by taking print output at high speed from the computer and passing it along at the much slower rate required by the printer. Print buffers vary in sophistication: some simply hold the next few characters to be printed, and others can queue, reprint, or delete documents sent for printing.

printed circuit board n. A flat board made of nonconducting material, such as plastic or fiberglass, on which chips and other electronic components are mounted, usually in predrilled holes designed to hold them. The component holes are connected electrically by predefined conductive metal pathways that are printed on the surface of the board. The metal leads protruding from the electronic components are soldered to the conductive metal pathways to form a connection. A printed circuit board should be held by the edges and protected from dirt and static electricity to avoid damage. See the illustration. Acronym: PCB.





Printed circuit board.

privileged mode Prodigy

privileged mode *n*. A mode of execution, supported by the protected mode of the Intel 80286 and higher microprocessors, in which software can carry out restricted operations that manipulate critical components of the system, such as memory and input/output ports (channels). Application programs cannot be executed in privileged mode; the heart (kernel) of the OS/2 operating system can be, as can the programs (device drivers) that control devices attached to the system.

privileges n. See access privileges.

PRN *n*. The logical device name for *printer*. A name reserved by the MS-DOS operating system for the standard print device. PRN usually refers to a system's first parallel port, also known as LPT1.

.pro *n*. One of seven new top-level domain names approved in 2000 by the Internet Corporation for Assigned Names and Numbers (ICANN), pro is meant for use in Web sites relating to professions such as physicians, accountants, and lawyers. Six of the new domains became available for use in the spring of 2001; negotiations are still underway for the final registry agreement for the pro domain.

probability n. The likelihood that an event will happen, which can often be estimated mathematically. In mathematics, statistics and probability theory are related fields. In computing, probability is used to determine the likelihood of failure or error in a system or device.

problem solving *n*. **1.** The process of devising and implementing a strategy for finding a solution or for transforming a less desirable condition into a more desirable one. **2.** An aspect of artificial intelligence wherein the task of problem solving is performed solely by a program. *See also* artificial intelligence.

procedural language *n*. A programming language in which the basic programming element is the procedure (a named sequence of statements, such as a routine, subroutine, or function). The most widely used high-level languages (C, Pascal, Basic, FORTRAN, COBOL, Ada) are all procedural languages. *See also* procedure. *Compare* nonprocedural language.

procedural rendering *n*. The rendering of a two-dimensional image from three-dimensional coordinates with texturing according to user-specified conditions, such as direction and degree of lighting.

procedure *n*. In a program, a named sequence of statements, often with associated constants, data types, and variables, that usually performs a single task. A procedure can usually be called (executed) by other procedures, as well as by the main body of the program. Some languages distinguish between a procedure and a function, with the latter (the function) returning a value. *See also* function, parameter, procedural language, routine, subroutine.

procedure call *n*. In programming, an instruction that causes a procedure to be executed. A procedure call can be located in another procedure or in the main body of the program. *See also* procedure.

process¹ n. A program or part of a program; a coherent sequence of steps undertaken by a program.

process² vb. To manipulate data with a program.

process-bound *adj*. Limited in performance by processing requirements. *See also* computation-bound.

process color n. A method of handling color in a document in which each block of color is separated into its subtractive primary color components for printing: cyan, magenta, and yellow (as well as black). All other colors are created by blending layers of various sizes of halftone spots printed in cyan, magenta, and yellow to create the image, See also color model, color separation (definition 1). Compare spot color.

processing *n*. The manipulation of data within a computer system. Processing is the vital step between receiving data (input) and producing results (output)—the task for which computers are designed.

processor n. See central processing unit, microprocessor.

Processor Direct Slot n. See PDS (definition 1).

Processor Input/Output n. See PIO.

Procmall *n*. An open-source e-mail-processing utility for Linux and other UNIX-based computers and networks. Procmail can be used to create mail servers and mailing lists, filter mail, sort incoming mail, preprocess mail, and perform other mail-related functions.

Prodlgy *n*. An Internet service provider (ISP) that offers Internet access and a wide range of related services. Prodigy was founded by IBM and Sears as a proprietary online service, was acquired by International Wireless in 1996, and in 1999 entered into a partnership with SBC Commu-



nications. The addition of SBC's Internet customer base made Prodigy the third largest ISP in the United States.

Prodlgy Information Service *n*. An online information service founded by IBM and Sears. Like its competitors America Online and CompuServe, Prodigy offers access to databases and file libraries, online chat, special interest groups, e-mail, and Internet connectivity. *Also called:* Prodigy.

product *n*. 1. An operator in the relational algebra used in database management that, when applied to two existing relations (tables), results in the creation of a new table containing all possible ordered concatenations (combinations) of tuples (rows) from the first relation with tuples from the second. The number of rows in the resulting relation is the product of the number of rows in the two source relations. *Also called:* Cartesian product. *Compare* inner join. 2. In mathematics, the result of multiplying two or more numbers. 3. In the most general sense, an entity conceived and developed for the purpose of competing in a commercial market. Although computers are products, the term is more commonly applied to software, peripherals, and accessories in the computing arena.

production system *n*. In expert systems, an approach to problem solving based on an "IF this, THEN that" approach that uses a set of rules, a database of information, and a "rule interpreter" to match premises with facts and form a conclusion. Production systems are also known as rule-based systems or inference systems. *See also* expert system.

Professional Graphics Adapter *n.* A video adapter introduced by IBM, primarily for CAD applications. The Professional Graphics Adapter is capable of displaying 256 colors, with a horizontal resolution of 640 pixels and a vertical resolution of 480 pixels. *Acronym:* PGA.

Professional Graphics Display *n*. An analog display introduced by IBM, intended for use with their Professional Graphics Adapter. *See also* Professional Graphics Adapter.

profile¹ n. See user profile.

profile² vb. To analyze a program to determine how much time is spent in different parts of the program during execution.

profiler *n*. A diagnostic tool for analyzing the run-time behavior of programs.

Profiles for Open Systems Internetworking Technology n. See POSIT.

program¹ n. A sequence of instructions that can be executed by a computer. The term can refer to the original source code or to the executable (machine language) version. Also called: software. See also program creation, routine, statement.

program² vb. To create a computer program, a set of instructions that a computer or other device executes to perform a series of actions or a particular type of work.

program button *n*. On a handheld device, a navigation control that is pressed to launch an application, *Also called*: application button.

program card n. See PC Card, ROM card.

program cartridge n. See ROM cartridge.

program comprehension tool *n*. A software engineering tool that facilitates the process of understanding the structure and/or functionality of computer applications. *Acronym:* PCT. *Also called:* software exploration tool.

program counter n. A register (small, high-speed memory circuit within a microprocessor) that contains the address (location) of the instruction to be executed next in the program sequence.

program creation n. The process of producing an executable file. Traditionally, program creation comprises three steps: (1) compiling the high-level source code into assembly language source code; (2) assembling the assembly language source code into machine-code object files; and (3) linking the machine-code object files with various data files, run-time files, and library files into an executable file. Some compilers go directly from high-level source to machine-code object, and some integrated development environments compress all three steps into a single command. See also assembler, compiler (definition 2), linker, program.

program encapsulation *n*. A method of dealing with programs with Year 2000 problems that entailed modifying the data with which a program worked. The input data is modified to reflect a parallel date in the past that the program can handle. When output is generated, that data is changed again, to reflect the correct date. The program itself remains unchanged.

program file n. A disk file that contains the executable portions of a computer program. Depending on its size and

remote monitoring replace

net, remote login is done primarily by rlogin and telnet. See also rlogin¹ (definition 1), telnet¹.

remote monitoring n. See RMON.

remote network monitoring n. See RMON.

Remote PC n. See remote system.

remote procedure call *n*. In programming, a call by one program to a second program on a remote system. The second program generally performs a task and returns the results of that task to the first program. *Acronym:* RPC.

remote system *n*. The computer or network that a remote user is accessing via a modem. *See also* remote access. *Compare* remote terminal.

remote terminal *n*. A terminal that is located at a site removed from the computer to which it is attached. Remote terminals rely on modems and telephone lines to communicate with the host computer. *See also* remote access. *Compare* remote system.

removable disk *n*. A disk that can be removed from a disk drive. Floppy disks are removable; hard disks usually are not. *Also called:* exchangeable disk.

REM statement *n*. Short for remark statement. A statement in the Basic programming language and the MS-DOS and OS/2 batch file languages that is used to add comments to a program or batch file. Any statement beginning with the word *REM* is ignored by the interpreter or compiler or the command processor. *See also* comment.

rename *n*. A command in most file transfer protocol (FTP) clients and in many other systems that allows the user to assign a new name to a file or files.

render *vb*. To produce a graphic image from a data file on an output device such as a video display or printer.

rendering *n*. The creation of an image containing geometric models, using color and shading to give the image a realistic look. Usually part of a geometric modeling package such as a CAD program, rendering uses mathematics to describe the location of a light source in relation to the object and to calculate the way in which the light would create highlights, shading, and variations in color. The degree of realism can range from opaque, shaded polygons to images approximating photographs in their complexity. *See also* ray tracing.

RenderMan Shading Language n. A C-like graphics and rendering language developed by Pixar.

repaginate vb. To recalculate the page breaks in a document.

Repeat *n*. A command in Microsoft Word that causes all information contained in either the last command dialog box or the last uninterrupted editing session to be repeated.

repeat counter *n*. A loop counter; typically, a register that holds a number representing how many times a repetitive process has been or is to be executed.

Repeat delay n. A delay for the amount of time that elapses before a character begins repeating when you hold down a key.

repeater *n*. A device used on communications circuits that decreases distortion by amplifying or regenerating a signal so that it can be transmitted onward in its original strength and form. On a network, a repeater connects two networks or two network segments at the physical layer of the ISO/OSI reference model and regenerates the signal.

repeating Ethernet n. See repeater.

repeat key *n*. On some keyboards, a key that must be held down at the same time as a character key to cause the character key's key code to be sent repeatedly. On most computer keyboards, however, a repeat key is not needed because a key automatically repeats if held down for longer than a brief delay. *Compare* typematic.

RepeatKeys *n*. A feature of Windows 9x and Windows NT that allows a user to adjust or disable the typematic keyboard feature so as to accommodate users with restricted mobility, who may activate typematic by accident because they have trouble lifting their fingers from the keys. *See also* typematic. *Compare* BounceKeys, FilterKeys, MouseKeys, ShowSounds, SoundSentry, StickyKeys, ToggleKeys.

repetitive strain injury n. An occupational disorder of the tendons, ligaments, and nerves caused by the cumulative effects of prolonged repetitious movements. Repetitive strain injuries are appearing with increasing frequency among office workers who spend long hours typing at computerized workstations that are not equipped with safeguards such as wrist supports. Acronym: RSI. See also carpal tunnel syndrome, ergonomic keyboard, wrist support.

replace *vb.* To put new data in the place of other data, usually after conducting a search for the data to be replaced. Text-based applications such as word processors typically include search-and-replace commands. In such

R

Xerox PARC XML-RPC

Xerox PARC n. Short for **Xerox** Palo Alto Research Center. Xerox's research and development facility in Palo Alto, California. Xerox PARC is the birthplace of such innovations as the local area network (LAN), the laser printer, and the graphical user interface (GUI).

XFCN *n*. Short for external function. An external code resource that returns a value after it has completed executing. XFCNs are used in HyperCard, a hypermedia program developed for the Macintosh. *See also* HyperCard, XCMD.

XFDL *n*. Short for Extensible Forms Description Language, a document description language introduced and submitted to the World Wide Web Committee in 1998 by the Canadian Internet forms company UWI.Com. XFDL is an XML-based language for describing complex forms, such as legal and government documents. It is designed to allow for interactivity yet remain consistent with Internet standards.

XGA n. See Extended Graphics Array.

x-height *n*. In typography, the height of the lowercase letter x in a particular font. The x-height thus represents the height of the body only of a lowercase letter, excluding ascenders (such as the top of the letter b) and descenders (such as the tail on the letter g). *See also* ascender, descender.

XHTML *n*. Short for Extensible Hypertext Markup Language. A markup language incorporating elements of HTML and XML. Web sites designed using XHTML can be more readily displayed on handheld computers and digital phones equipped with microbrowsers. XHTML was released for comments by the World Wide Web Consortium (W3C) in September 1999. *See also* HTML, microbrowser, XML.

XIP *n*. See execute in place.

XLANG n. A derivative XML language that describes the logical sequencing of business processes, as well as the implementation of the business process by using various application services.

XLink *n*. An XML language that provides a set of attributes that are used to create links between resources. XLink provides complex extended linking, link behavior, and management capabilities. XLink is able to describe links that connect sets of resources, point to multiple targets, or serve multiple roles within an XML document.

XLL *n*. Acronym for eXtensible Linking Language. Broad term intended to denote the family of XML linking/pointing/addressing languages, which include XLink, XPointer, and XPath.

XMI n. 1. Acronym for **XML** Metadata Interchange Format. An object-based model for exchanging program data across the Internet. XMI is sponsored by IBM, Unisys, and others and was submitted as a proposed standard to the Object Management Group (OMG); it is now one of OMG's recommended technologies. XMI is designed to allow for storing and sharing programming information and exchanging data among tools, applications, and storage locations through a network or the Internet so that software developers can collaborate on applications, even if they are not all using the same development tools. 2. As XMI bus, a 64-bit parallel bus supported on certain DEC and Alpha-Server processors. An XMI bus is capable of transferring data, exclusive of addressing overhead, at 100 Mbps.

XML *n*. Acronym for eXtensible Markup Language, a condensed form of SGML (Standard Generalized Markup Language), XML lets Web developers and designers create customized tags that offer greater flexibility in organizing and presenting information than is possible with the older HTML document coding system. XML is defined as a language standard published by the W3C and supported by the industry. *See also* SGML.

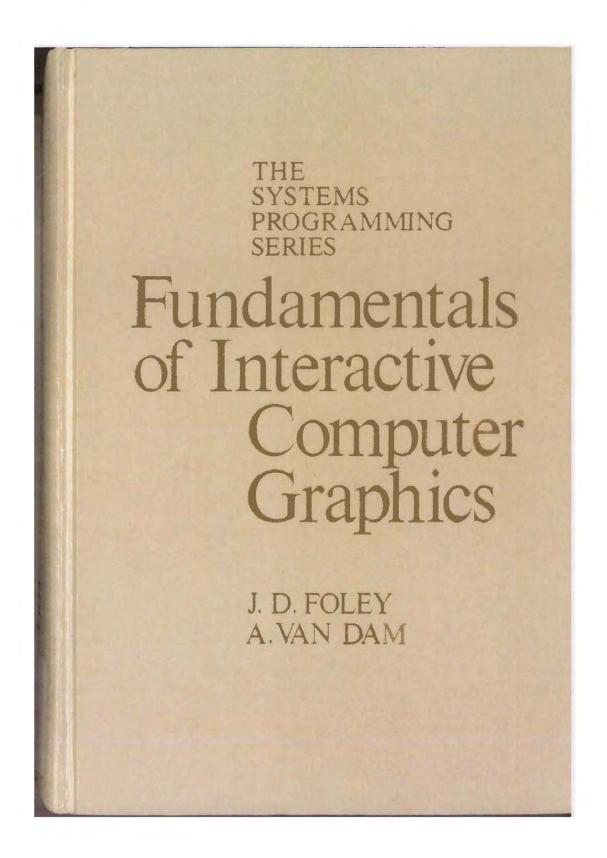
XML attribute *n*. Information added to a tag to provide more information about the tag, such as <ingredient quantity="2"units="cups">flour</ingredient>.

XML element *n*. Information delimited by a start tag and an end tag in an eXtensible Markup Language (XML) document. An example would be <Lastname> Davalio</LastName>.

XML entities *n*. Combinations of characters and symbols that replace other characters when an XML document is parsed, usually those that have other meanings in XML. For example, & t; represents the & lt; symbol, which is also the opening bracket for a tag.

XML Metadata Interchange Format n. See XMI (definition 1).

XML-RPC *n.* Acronym for eXtensible Markup Language-Remote Procedure Call. A set of XML-based implementations that allows cross-platform and cross-programming language procedure calls over the Internet. XML-RPC



ATI 2086 LG v. ATI IPR2015-00326

Fundamentals of Interactive Computer Graphics

JAMES D. FOLEY
The George Washington University

ANDRIES VAN DAM

Brown University



Addison-Wesley Publishing Company

Reading, Massachusetts • Menlo Park, California London • Amsterdam • Don Mills, Ontario • Sydney Sponsoring Editor: William B. Gruener

Production Editor: Rima Zolina Designer: Herb Caswell Illustration: ANCO/Boston

Cover Design: Richard Hannus Art coordinator: Dick Morton

This book is in the Addison-Wesley Systems Programming Series Consulting editors: IBM Editorial Board

Library of Congress Cataloging in Publication Data

Foley, James D 1942-

Fundamentals of interactive computer graphics.

(The Systems programming series)

Bibliography: p.

Includes index.

1. Computer graphics. 2. Interactive computer

systems. I. Van Dam, Andries, 1938- joint author.

II. Title.

T385.F63 001.64'43

80-24311

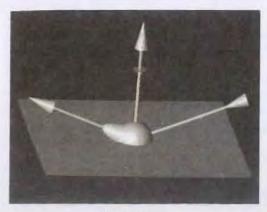
SBN 0-201-14468-9

Reprinted with corrections, July 1984

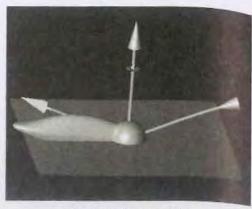
Copyright © 1982 by Addison-Wesley Publishing Company, Inc. Philippines copyright 1982 by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada. Library of Congress Catalog Card No. 80-24311

ISBN: 0-201-14468-9 NOPQR-HA-8987



(a) Phong model

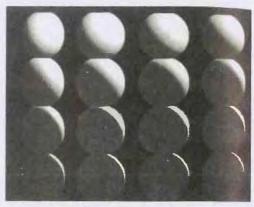


(b) Torrance-Sparrow model

Fig. 16.5 Comparison of Phong and Torrance-Sparrow models for a 70° angle of incident light (by J. Blinn [BLIN77a], courtesy University of Utah).



(a) Phong model



(b) Torrance-Sparrow model

Fig. 16.6 Comparison of the Phong and Torrance-Sparrow models for a metallic sphere illuminated by a light source from different directions. Differences are most apparent for back-lit cases (bottom rows) (by J. Blinn [BLIN77a], courtesy University of Utah).

16.4 POLYGON MESH SHADING

There are three basic ways to shade objects defined by polygon meshes. In order of increasing complexity, they are: constant shading, intensity interpolation shading, and normal-vector interpolation shading. In each case, any of the shading models from the previous two sections can be used. Recall that color shading just involves three equations rather than one.

Constant shading calculates a single intensity value for shading an entire polygon. Several assumptions are made:

- 1. The light source is at infinity, so $\vec{N} \cdot \vec{L}$ is constant across the polygonal face;
- 2. The viewer is at infinity, so $\overline{N} \cdot \overline{V}$ is constant across the polygon face;
- 3. The polygon represents the actual surface being modeled, and is not an approximation to a curved surface.

If either of the first two assumptions is unacceptable, then an average \overline{L} and \overline{V} might be used, perhaps calculated at the center of the polygon.

The final assumption is most often the one which is incorrect and has a much more substantial effect on the resulting image than the other two. The effect is that each visible polygonal facet of the approximated surface is distinguishable, because each is a slightly different intensity than its neighbors. The difference in shading on adjacent facets is accentuated by the Mach band effect, which was discovered in 1865 by E. Mach and is described in detail in [RATL65]. The effect is one of exaggeration of intensity change at any edge where there is a discontinuity in magnitude or slope of intensity. Figure 16.7 shows, for two separate cases, the actual and perceived changes in intensity along a surface. The effect is caused by lateral inhibition of the receptors in the eye, whose response to light is influenced by adjacent receptors in inverse relation to the distance to the adjacent receptor. Receptors immediately to the brighter side of an intensity change have more response than those further from the edge, because they receive less inhibition from their neighbors on the darker side. Similarly, receptors immediately to the darker side of an intensity change will have less response than those further into the darker area, because they receive more inhibition from their neighbors on the brighter side.

Figure 16.8(b) shows a car with constant shading. The Mach band effect is quite evident. Even though the polygonal patches are quite noticeable, the image is much more realistic than that in Fig. 16.8(a), which shows only the polygon edges.

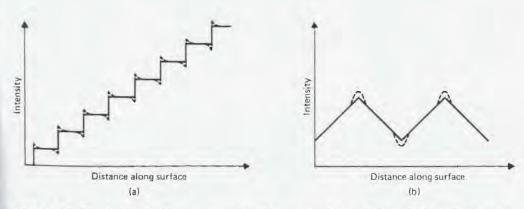
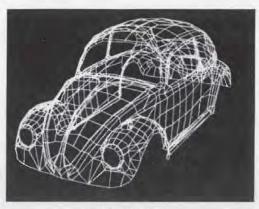


Fig. 16.7 Mach band effect—actual and perceived intensities: dashed lines—perceived intensity; solid lines—actual intensity.



(a) Polygon outlines



(b) Constant shading

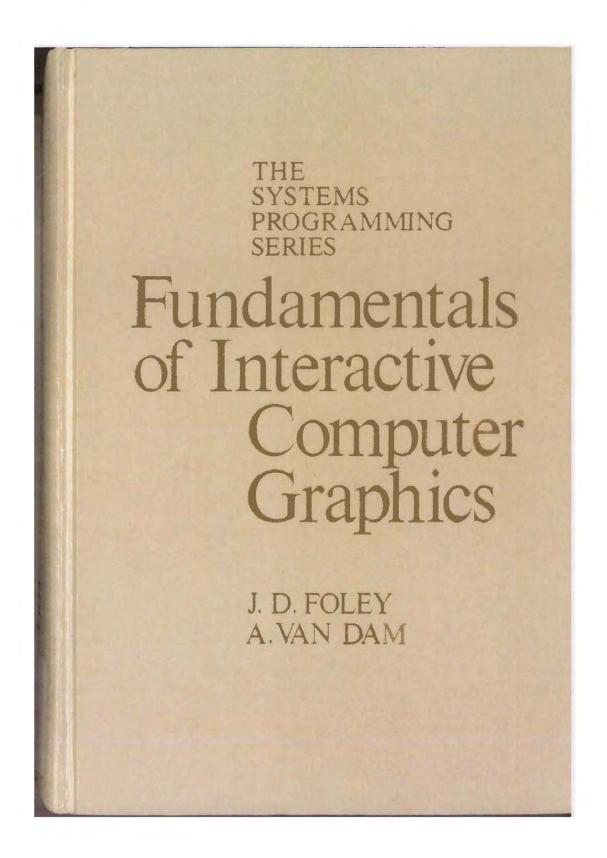


(c) Gouraud shading

Fig. 16.8 Car body displayed three ways (courtesy University of Utah).

Intensity interpolation shading, usually known from the name of its developer as Gouraud shading [GOUR71], eliminates intensity discontinuities. Figure 16.8(c) shows a Gouraud-shaded car. The intensity ridge running down the hood on the right side of the picture, close to the fender, is a Mach band caused by a rapid change in the slope of the intensity curve: Gouraud shading does not completely eliminate such intensity changes.

The Gouraud shading process consists of four steps. First, surface normals are calculated. Second, *vertex normals* are calculated by averaging the surface normals of all polygonal facets that share the vertex (Fig. 16.9). If an edge is meant to be visible (such as at the joint between a planes' wing and body), then two vertex normals, one for each side of the edge, are found by separately averaging the normals of polygons on each side of the edge. Third, *vertex intensities* are found by using the vertex normals with any desired shading model. Finally, each polygon is shaded by linear interpolation of vertex intensities along each edge and then between edges along each scan line (Fig. 16.10).



ATI 2086 LG v. ATI IPR2015-00326

Fundamentals of Interactive Computer Graphics

JAMES D. FOLEY
The George Washington University

ANDRIES VAN DAM

Brown University



Addison-Wesley Publishing Company

Reading, Massachusetts • Menlo Park, California London • Amsterdam • Don Mills, Ontario • Sydney Sponsoring Editor: William B. Gruener

Production Editor: Rima Zolina Designer: Herb Caswell Illustration: ANCO/Boston

Cover Design: Richard Hannus Art coordinator: Dick Morton

This book is in the Addison-Wesley Systems Programming Series Consulting editors: IBM Editorial Board

Library of Congress Cataloging in Publication Data

Foley, James D 1942-

Fundamentals of interactive computer graphics.

(The Systems programming series)

Bibliography: p.

Includes index.

1. Computer graphics. 2. Interactive computer

systems. I. Van Dam, Andries, 1938- joint author.

II. Title.

T385.F63 001.64

001.64'43 80-24311

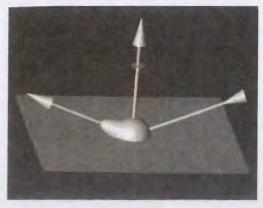
SBN 0-201-14468-9

Reprinted with corrections, July 1984

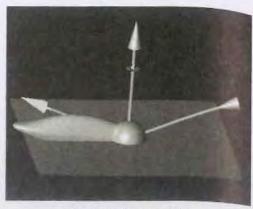
Copyright © 1982 by Addison-Wesley Publishing Company, Inc. Philippines copyright 1982 by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada. Library of Congress Catalog Card No. 80-24311

ISBN: 0-201-14468-9 NOPQR-HA-8987



(a) Phong model



(b) Torrance-Sparrow model

Fig. 16.5 Comparison of Phong and Torrance-Sparrow models for a 70° angle of incident light (by J. Blinn [BLIN77a], courtesy University of Utah).



(a) Phong model



(b) Torrance-Sparrow model

Fig. 16.6 Comparison of the Phong and Torrance-Sparrow models for a metallic sphere illuminated by a light source from different directions. Differences are most apparent for back-lit cases (bottom rows) (by J. Blinn [BLIN77a], courtesy University of Utah).

16.4 POLYGON MESH SHADING

There are three basic ways to shade objects defined by polygon meshes. In order of increasing complexity, they are: constant shading, intensity interpolation shading, and normal-vector interpolation shading. In each case, any of the shading models from the previous two sections can be used. Recall that color shading just involves three equations rather than one.

Constant shading calculates a single intensity value for shading an entire polygon. Several assumptions are made:

- 1. The light source is at infinity, so $\vec{N} \cdot \vec{L}$ is constant across the polygonal face;
- 2. The viewer is at infinity, so $\overline{N} \cdot \overline{V}$ is constant across the polygon face;
- 3. The polygon represents the actual surface being modeled, and is not an approximation to a curved surface.

If either of the first two assumptions is unacceptable, then an average \overline{L} and \overline{V} might be used, perhaps calculated at the center of the polygon.

The final assumption is most often the one which is incorrect and has a much more substantial effect on the resulting image than the other two. The effect is that each visible polygonal facet of the approximated surface is distinguishable, because each is a slightly different intensity than its neighbors. The difference in shading on adjacent facets is accentuated by the Mach band effect, which was discovered in 1865 by E. Mach and is described in detail in [RATL65]. The effect is one of exaggeration of intensity change at any edge where there is a discontinuity in magnitude or slope of intensity. Figure 16.7 shows, for two separate cases, the actual and perceived changes in intensity along a surface. The effect is caused by lateral inhibition of the receptors in the eye, whose response to light is influenced by adjacent receptors in inverse relation to the distance to the adjacent receptor. Receptors immediately to the brighter side of an intensity change have more response than those further from the edge, because they receive less inhibition from their neighbors on the darker side. Similarly, receptors immediately to the darker side of an intensity change will have less response than those further into the darker area, because they receive more inhibition from their neighbors on the brighter side.

Figure 16.8(b) shows a car with constant shading. The Mach band effect is quite evident. Even though the polygonal patches are quite noticeable, the image is much more realistic than that in Fig. 16.8(a), which shows only the polygon edges.

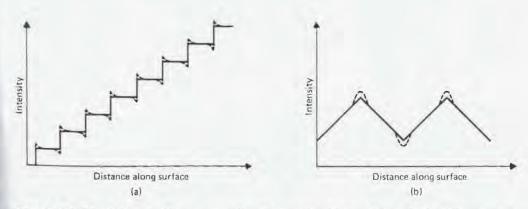
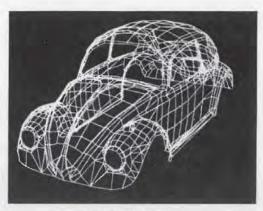


Fig. 16.7 Mach band effect—actual and perceived intensities: dashed lines—perceived intensity; solid lines—actual intensity.



(a) Polygon outlines



(b) Constant shading



(c) Gouraud shading

Fig. 16.8 Car body displayed three ways (courtesy University of Utah).

Intensity interpolation shading, usually known from the name of its developer as Gouraud shading [GOUR71], eliminates intensity discontinuities. Figure 16.8(c) shows a Gouraud-shaded car. The intensity ridge running down the hood on the right side of the picture, close to the fender, is a Mach band caused by a rapid change in the slope of the intensity curve: Gouraud shading does not completely eliminate such intensity changes.

The Gouraud shading process consists of four steps. First, surface normals are calculated. Second, *vertex normals* are calculated by averaging the surface normals of all polygonal facets that share the vertex (Fig. 16.9). If an edge is meant to be visible (such as at the joint between a planes' wing and body), then two vertex normals, one for each side of the edge, are found by separately averaging the normals of polygons on each side of the edge. Third, *vertex intensities* are found by using the vertex normals with any desired shading model. Finally, each polygon is shaded by linear interpolation of vertex intensities along each edge and then between edges along each scan line (Fig. 16.10).



DirectX 10 Architecture

for Chrome 400 Series Discrete Graphics Processors

> A S3 Graphics White Paper

> > ATI 2087 LG v. ATI IPR2015-00326

WP016-A.0 7/21/2007

© 2007 S3 Graphics. All rights reserved. www.s3graphics.com

0001



Introduction

This White Paper provides an overview of Microsoft's DirectX 10 architecture. DirectX 10 compatible features provide an important component of the enhanced experience available to users of systems featuring S3 Graphics Chrome 400 Series graphics processors. These processors are designed especially for Microsoft DirectX 10 and Windows Vista.

DirectX 3D is the standard API (application programming interface) that allows graphics hardware to render and support graphics on Microsoft Windows platforms. This API is a common interface or middleware that provides a hardware abstraction layer which allows developers of an application, such as a 3D game or CAD program, to access the graphics hardware via programming calls to the operating system (OS). When the application makes a request to draw an image on the screen, the API calls the OS, which in turn will invoke the graphics processor (GPU) driver to communicate with the graphics hardware to draw the corresponding image and output the result to the display. By using the standard DirectX interface, application developers need only need be concerned about their specific application, without needing to be concerned about details of the underlying hardware implementation. This allows developers to quickly create many visually stunning images and realistic detail, by providing fast access to the advanced hardware capabilities of today's leading edge GPUs.

Previous generations of Microsoft's DirectX (DX) 3D, had significant changes. Fixed function hardware units were used in DirectX generations up to DX7. Then programmable hardware shader units with new user-defined programming capabilities appeared for DX8. DX9 featured added hardware functionality and programmability. The latest release from Microsoft is DX10, which introduces a new architecture that is the subject of this white paper. S3 Graphics continues to work closely with Microsoft's DirectX team to extend its leadership in graphics technology with high performance parts, including the Chrome S20 Series processors based on DX9.0c, and the Chrome 400 Series graphics processor line with advanced DX10 support.

Microsoft's latest DirectX release, DX10, extends the API beyond the limitations of previous generations. The DX10 API is the first redesign to the underlying architecture of DirectX 3D. New are optimized run-time features, CPU off-loading during state changes, a hardware geometry shader, texture arrays, and other graphics rendering enhancements that allow cinematic-like image quality. S3 Graphics Chrome 400 Series product line fully supports these new capabilities of DX10.

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 3 of 19

DirectX 9 Hardware Pipeline

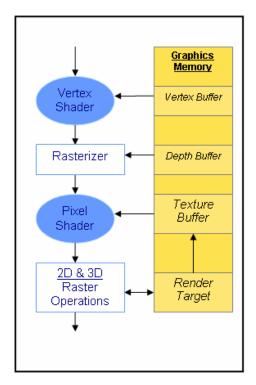


Figure 1. DirectX 8/9 Graphics Pipeline

The DirectX 8/9 pipeline is diagrammed in the above figure, which shows the basic features of a DX8/9 GPU. With the introduction of DX8/9 GPUs such as S3 Graphics' Chrome 20 Series, the transition from a fixed to a programmable pipeline changed the graphics landscape. Fixed function pipelines meant the hardware blocks were hard-coded with specific graphics algorithms. Application developers had to limit their development to what the hardware supported. DirectX 8 introduced and DirectX 9 expanded programmable pipelines which provided an additional programmable API layer closer to the graphics hardware. Developers can take advantage of this layer by using shader assembly language to creatively write specific code to control the different shaders and elements of the programmable pipeline. The main parts of the pipeline are as follows:

Vertex shaders (VS) replace the "Transform and lighting engine" logic prevalent in
previous generations of graphics hardware. The VS can only manipulate vertices and
transform the shapes of objects from the 3D model space to be displayed on a 2D
screen. The VS also does per-vertex lighting based on computed color to give the vertex
more detail. The VS cannot create or destroy any vertices and the unit can only work on
one vertex at a time (in the API level). Actual graphics hardware may process batches or
packets of vertices in parallel to increase throughput.

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 4 of 19

- Rasterization is the process of mapping a triangle from object to image space (combining vertices from the VS output) and determining which screen pixels cover the triangle. All pixels inside the triangle are tested for visibility using the depth buffer and are kept if the triangle being rasterized is closer from a viewer perspective than other triangles. All invisible triangles and pixels are discarded since they will not be seen onscreen. This step prepares the object to be modified at the pixel level by the pixel shader.
- **Pixel shaders (PS)** calculate color and texture on each individual pixel. They give flexibility to developers by allowing high quality details to be shown on each object.
- The 2D and 3D **Raster Operation Pipeline** (ROP) is responsible for outputting the rendered object to the render target buffer from the pipeline after textures and blending have been applied.
- The graphics memory stores vertex and texture data in addition to the final object and frame that is to be drawn onscreen.

Limitations of DirectX 8/9

Microsoft designed DirectX 10 to address some of the following key disadvantages identified in the DirectX 8/9 architectures. These include the following.

1. API overhead is high for DX8/9

- When a DX9 application requires use of the graphics hardware to draw an object onscreen, the application needs to perform a call to the DX8/9 API to tell the OS what to do. The OS would then call the graphics driver, which would instruct the graphics hardware to perform the assigned task for the application.
- The DX8/9 API runtime provides resource management like allocation, virtualization, and initialization for the graphics hardware for vertex buffers, texture maps, and state changes. With the introduction of programmable shaders, runtime allocation tends to be harder to manage since there are more levels of abstraction, control and detail per scene.
- As Figure 2 illustrates, all DX8/9 functional or runtime calls from an application to the graphics hardware were done by the CPU (once per object), causing CPU bottlenecks whenever many objects need to be rendered for the current frame. The high API overhead also limits the number of objects per scene, causing potential loss of detail in each frame.
- State changes within the GPU for the shaders and textures generate additional overhead as the CPU had to decode state change instructions in order to implement visual details for realistic rendering to object surfaces and textures. If multiple visual effects need to be performed on an object, multiple passes through the hardware could be required. That translates into multiple state changes per pass performed by the CPU.
- The CPU overhead required to direct the GPU, in essence, became the bottleneck between the DX8/9 application and DX8/9 hardware, which limits the overall capacity for creating stunning visuals.

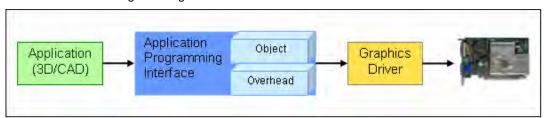


Figure 2. DirectX 8/9 Graphics API Interface

2. Vendor Variations in DirectX features

• Differences in DirectX feature support across GPU vendors, and even within a vendor's product lines, frequently cause problems for application development, because of the numerous levels of support that must be implemented across these multiple hardware platforms. Typical issues that must be addressed include allowing for the resource limitations of different hardware. The lack of support for optional features may prevent an application from running at its highest level on a particular platform. Differences in arithmetic precision in the hardware shaders, instruction and data types, and variations in storage of intermediate data formats may all affect the rendering process.

3. Hardware resource limitations

- The number of separate VS and PS units in the GPU are fixed. Applications requiring heavy use of one type of shader will cause the other shader to be idle. For example, large triangles create heavy loading for the PS, idling the VS. Since there is the possibility of an application being shader-limited, throughput of the rendering pipeline will be limited by the number and type of shaders. The DirectX 10 API allows graphics hardware to overcome this resource bottleneck by introducing a novel architecture which has been incorporated into all S3 Graphics Chrome 400 Series graphics processors.
- Another drawback of DX8/9 generation GPU architectures is their specialized focus
 on one task, that of 3D graphics rendering. A DirectX 8/9 GPU was designed to
 optimize that task, which in effect limited its ability to perform additional computing
 tasks. With the introduction of GPU architecture designed for DirectX 10, the GPU
 can now take on additional capabilities and complexity. The processor now becomes
 more of a general compute processor, capable of offloading the CPU from some
 basic tasks, and thus takes a significant step towards becoming a general purpose
 GPU (GPGPU).

DirectX 10 New Capabilities and Benefits

The main objective for redesigning the entire DirectX 10 API and hardware architecture was to provide a solution for the CPU overhead problem and the hardware capability issues. This re-design also incorporated benefits from the application development, API, and hardware perspectives. The improvements based on this extensive research and re-design are as follows:

1. Efficient runtime (lower API overhead)

- In DirectX 10 the number of possible states that need to be tracked by the system has been reduced, minimizing the overhead related to state changes.
- Overhead per object has been reduced which allows more objects per frame. This
 produces better graphic realism and a higher level of detail than was possible with
 previous generations of the API.
- The validation of objects has been redefined and the process is now more efficient. Validation checks the format of commands and the integrity of data sent by the application to make sure there are no interoperability issues with the hardware. The drawback of validation is a large CPU overhead at runtime. DX10 uses this feature minimally by only validating each object once when it is created, rather than every time the object is used, as was the case in DX9.

2. Reduced CPU loading

Rendering an object or applying multiple textures to an object in a repeated manner uses up valuable CPU cycles and overhead. DX10 has introduced several new instructions and hardware capabilities to help overcome rendering limitations.

- A new 3D pipeline unit called the geometry shader (GS) has been introduced with this iteration of DirectX. The GS can modify, create, or destroy primitive vertex data from the VS without CPU intervention, so no resource-intensive state changes or associated overhead is required by the API. In the past, any changes to the vertex data needed CPU-GPU coordination and state changes.
- The new hardware model in the DX10 pipeline gives more capability to the GPU to handle state changes and instructions. The DX10 GPU now includes built-in arithmetic and flow control logic, thereby providing flexibility in primitive shading and state change handling, and offloading the tasks once performed by the CPU.
- Stream out is a new feature that allows the VS/GS to output data directly into graphics memory where the data can be accessed automatically and repeatedly by the shader units. This is a great new feature controlled entirely by the GPU (with no CPU overhead), for recursive rendering on objects that require multiple passes through the pipeline. In addition, data from any step in the pipeline can go directly to memory. By avoiding the need to send data completely through the pipeline, resources are not wasted on processing intermediate vertices or pixels.
- Arrayed resources allow texture maps to be created as a linear array of up to 512 elements. Developers now have index instructions to access elements within the array in a single pass, so the GPU can work on multiple elements without any static switching overhead. For example, an environmental cube map can be stored in an

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 8 of 19

- array as six elements (one for each face of the cube), and the GPU can work on all six elements concurrently in one pass.
- **Multiple render targets** allow the GPU to create different versions of a scene in a single pass. DX10 has the ability to create up to eight render targets at a time.
- High dynamic-range rendering is another feature that brings realistic graphics rendering to the user experience. Formats used in the past to represent color in floating-point representation took at least twice the amount of storage compared to integer formats with half the precision. DX10 provides more efficient mechanism for storage of color components by providing floating point format RGB 11:11:10 (R/G 11-bits each, B 10-bits) and RGBE format (5-bit shared exponent for R/G/B with 9-bit mantissas for each). These formats allow a wider range of color and more vivid detail to be represented as seen in the examples below.

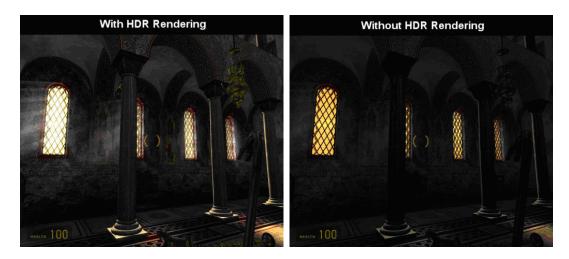


Figure 3. High Dynamic Range Rendering in Half-Life 2: Lost Coast

- A more complex method of utilizing occlusion query has also been implemented to conserve valuable GPU and CPU resources. Occlusion query is a method where non-visible primitives and objects in the Z-buffer are not rendered to save hardware computational resources. Because frame-to-frame rendering is very dynamic and implementation may vary with the application, occlusion query is not guaranteed to omit unseen pixels. DX10 takes it one step further by allowing the GPU to render complex objects in simple line drawing approximations. If the object needs to be drawn onscreen, the GPU already has the framework ready. If the object is not needed or is invisible onscreen, the GPU can throw away the object approximation, without wasting many CPU and GPU cycles.
- Data and resource mapping enhancements improve the ability of the GPU to access data in a timely manner. As an example, vertex buffer data for an application needs to be mapped to its memory address space, since that data can only be used by the application. The API and driver will allocate this buffer space at runtime either from the graphics memory (frame buffer) or system memory. While access to frame buffer is almost instantaneous, access to system memory is many magnitudes slower because of the communication lag between the GPU and system memory via the chipset. In DX10 resources are mapped according to how frequently they are used

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 9 of 19 (reads/writes) with four resource classes: default, immutable, dynamic, and staging. Using these new resource classes, developers can optimize performance by putting frequently used data in the frame buffer to be closer to the GPU and putting seldom-used data in system memory.

3. Additional constructs to improve efficiency

In graphics, multiple iterations of textures and blending usually take place to produce realistic images, such as re-creating hair moving in the wind or the ripples on the surface of a lake. These multiple loops previously required state changes and extensive CPU work to be performed. With DX10, state objects and constant buffers are now available to manage multiple loops in the rendering pipeline and increase the range of processing that can be done in one pass.

- State objects define what the graphics pipeline units should do as an object travels through the pipeline. The state objects have information to tell the pipeline which textures to blend or to tell the GS to create more detail for specific vertices in a part of the rendered frame. DX10 handles all of these details by introducing five state object commands, and programmers can work using a high-level language, instead of low-level constructs where they would need to keep track of all the pipeline stage units. The commands InputLayout, Sampler, Rasterizer, DepthStencil, and Blend are performed in the GPU, with minimal CPU intervention for state changes.
- Constant buffers store large amounts of predefined values (data) for items in a scene, so the CPU or GPU does not have to keep track of those values constantly. Each buffer stores up to 4096 constants hold information such as camera view/projection and light source color/position/intensity. Since these items have update intervals which may be once per frame or once per object, doing several hundred of these constant updates one at a time required significant CPU overhead when done using DirectX 9 or earlier. In DX10, the constant buffer groups the constants based on frequency of use and does batch processing to update the constants, which significantly reduces CPU use and dependence.

4. DX10 hardware specification set

- In pre-DX10 revisions of the API, hardware vendors were able to provide capability bits to inform the system about what features were supported in hardware. DX10 has changed this scheme. DX10 binds the 3D hardware feature set with a DirectX version number, so consistency across all hardware vendors exists and identifies support for a set of same basic features. Implementation of these features is the key for differentiating the graphics quality seen across vendors. S3 Graphics Chrome 400 Series processors have proprietary design implementations that give them an edge over the competition in visual quality and rendering capability.
- The basic programmable and fixed function pipelines of the past have been redesigned or eliminated. New and powerful hardware is capable of extending the implementation scope beyond the limits of rendering-only applications to provide high-throughput computing implementations for physics and AI.

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 10 of 19 The additional capabilities available in DX10 hardware and software offload most of the runtime events associated with rendering an object from the CPU to the GPU. With new and even more powerful functional units, an expanded instruction set, a new architecture that streamlines the graphics pipeline, efficient memory access methods, and multi-pass rendering capability, DX10 has introduced an astounding ability to generate graphics realism into our computing lives today and in the immediate future.

S3 Graphics DirectX 10 Architecture for Chrome 400 Series Processors WP016-A.0 7/21/2007 Page 11 of 19

DX10 Pipeline Introduction

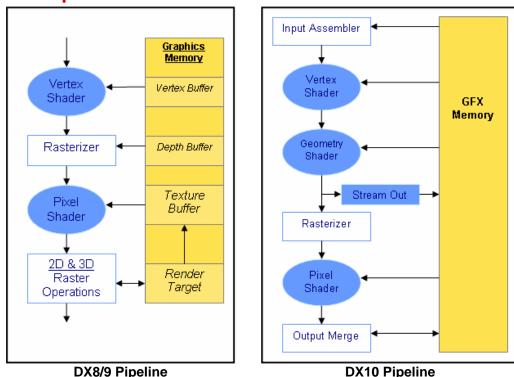


Figure 4. DX8/9 Pipeline Compared to DX10 Pipeline

Figure 4 shows a DX8/9 pipeline (left) compared to the latest DX10 hardware. Common functional units like the vertex/pixel shaders, rasterizer, and ROP/Output Merge (OM) block exist in both architectures and perform similar functions. The Input Assembler (IA) converts or replicates input vertex data from incoming streams (vertex structure) to be used by the pipeline. The key difference is the introduction of the Geometry Shader (GS) and Stream Output (SO) described below.

• The Geometry Shader (GS) takes the vertices of a primitive, such as a line, point, or triangle, and will either create additional vertices (generate data) or destroy the vertices. The GS increases the number of vertices by creating additional primitives composed of up to 1024 32-bit vertex data per instance. If a vertex is not needed, the GS can delete it from the rendering pipeline. The GS can also add additional elements to a primitive without needing to create a new vertex stream. In the past, the pipeline could not create or destroy vertices, only modify them. DX10 moves one step ahead by allowing even more flexibility and power in hardware, where the GS performs per-primitive modifications and also accesses adjacent primitive information. For example, in a GS implementation for a realistic shadow rendering, the GS can control a point or line and its neighboring primitive, as well as control the displacement mapping, where more detail can be shown with the creation of new primitives based on height maps.

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 12 of 19

- Stream output can write vertex or primitive information from the VS/GS to a stream buffer
 in memory immediately after the GS stage. In the past a primitive had to exit the PS and
 then it could be written to the render target buffer in memory. Now data in the stream
 buffer can be used recursively or iteratively by other functional blocks in the pipeline on
 an as-needed basis to improve data re-use efficiency. Other uses of this feature are
 physics calculation support such as used in particle systems where ongoing calculations
 are needed to generate and destroy primitives to simulate water, smoke, and clouds.
- Graphics memory has been changed from an area that stored vertex and texture data separately, to memory where each independent shader unit can access the same data. The data storage formats have also been updated to allow the pipeline to store and use multiple format types to increase flexibility. The memory can store data in arrays which allows recirculation of data and texture fetches by the VS, GS, and PS.

DX10 introduces a unified architecture that builds upon the pipeline diagramed in Figure 4. The DX10 pipeline combines three types of shaders into unified execution units capable of handling VS, GS, and PS instructions. The DX10 pipeline architectural design has solved many issues seen in previous generations of DirectX. It has added significant changes to the software, instruction set, instruction support/capability, as well as requiring new hardware blocks and features.

This new design still has a limitation in the utilization rate for each shader type based on the application (see Figure 5 below). If complex geometry calculations and processing are needed, then the VS may be fully utilized while the PS might remain idle. If an application is pixel processing heavy, then the PS may be running at full speed while the VS will only be partially loaded.

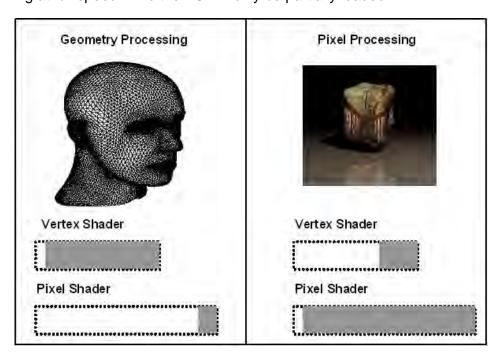
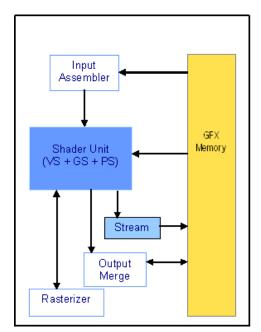


Figure 5. Vertex and Pixel Shader Resource Utilization for Different Applications

DX10 Shader Model 4.0

Shader Model 4.0 (SM4.0) is the new instruction set architecture (ISA) for DX10 that looks at the graphics in a unified way. Some keys advantages of SM4.0 are:

- Easy Programmability: Developers do not need to be bogged down with the low-level details of the hardware. In the past, programmers needed to control and write different low-level program code for each individual shader (VS/PS). Each shader was also considered an individual virtual machine that had separate input/output/general registers that had to be tracked for shader I/O, memory transfers, and intermediate data storage. SM4.0 instructions hide the low-level implementation details and incorporate all the pipeline flow control.
- Flexible Load Balancing: The new unified ISA allows flexibility. Developers can now look at these shaders units as one cohesive block (single common core virtual machine) instead of separate blocks, as shown in Figure 6. The unified shader is made up of shader blocks that can handle all vertex, pixel, and geometry instructions, so the GPU is fully utilized without concern for shader loading imbalances (geometry processing vs. pixel processing, as shown in Figure 5). There is also additional logic to load balance the shader units to keep all functional units fully utilized. If more pixel processing is needed, then more of the unified shader blocks can be allocated to pixel processing to increase throughput. The same shader-type allocation can be done with the VS and GS, as seen in Figure 7.
- Unified Shader Code: Developers code in "shader" instructions, not VS/PS/GS specific
 code.
- **Programmable Offloading:** The unified model helps offload the CPU state change overhead by incorporating flow control logic that programmers can control.



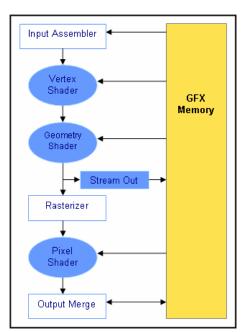


Figure 6. Unified Shader Model (Left) Compared to Basic DX10 Pipeline (Right)

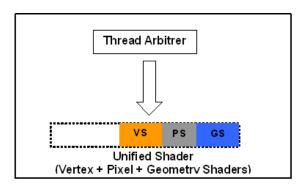


Figure 7. Unified Shader Utilized for Different Shader Types

 Additional resources, data formats, and instructions are available to the programmer for more efficient use and coupling of the graphics hardware to the application level.

Feature	DX9	DX10	
Instruction Slots	512	64K	
Constant Registers	256	4096 (x16)	
Temporary Registers	32	4096	
Render Targets	4	8	
Textures	16	128	
Texture Size	2K x 2K	8K x 8K	
Load Operations	No	Yes	
Sample Offsets	No	Yes	
Flow Control	Static/Dynamic	Dynamic	

Table 1. Basic Comparison Table of DX9 and DX10

- Full integer and bitwise instructions allow the GPU to compute complex algorithms more efficiently in integer format instead of converting between floating point and integer.
- Switch statements are another great addition because they provide multiple paths/options
 when rendering objects on a per-primitive basis. This means the GPU can replicate
 objects (instancing) and also provide unique characteristics for each object independently
 of the other objects in the scene.
- Increased texture support and size greatly enhance visual quality.
 - o In DX9, developers only had 16 textures to work with at a given time and their size limitation was 4096 x 4096. Since the application of multiple textures required multiple texture changes and multiple state changes (large CPU overhead), developers were limited in what they could do. If developers needed multiple textures, they created a texture atlas that combined many smaller textures that could be accessed by indexing into the atlas. This method proved very inefficient since the boundaries between smaller textures were not as clearly defined and the atlas could only hold a certain amount of textures so there was a trade-off between storing fewer (larger) textures or more (smaller) textures.
 - SM4.0 has new instructions and indexed texture arrays, which can store 512 textures with resolutions of up to 8192 x 8192. This method effectively replaces the texture atlas with a large array that can be indexed into easy-to-access multiple textures. In addition, the number of textures a shader can use has increased from 16 to 128, allowing hardware to take advantage of the texture array to add more detail to all objects in a frame.

S3 Graphics
DirectX 10 Architecture
for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 15 of 19

High-Level Shader Language (HLSL 10)

HLSL 10 is the name given to the programming language developers use to take advantage of DX10 shader and hardware capabilities. The advantages of HLSL are many.

- Application developers do not need to worry about using assembly-like instructions to control the shader and pipeline at the hardware level.
- · Applications can offload the task of resource management.
- Bind-by-name to bind-by-position allows less overhead at runtime. Bind-by-name in DX9 performs checks like matching input/output between hardware functional units and matching vertex buffer format with the vertex shader. Any type mismatch would cause huge overhead since the hardware was not as flexible. In DX10 shader units have associated signatures with their inputs and outputs. As long as the output of the preceding stage is compatible with the input of the following stage, then the data type mismatch is allowed since DX10 allows multiple data formats to be used at any stage in the pipeline.
- DX10 has a "view" method for representing resources such as vertex buffers or texture maps, which can be read in many different formats, so that they are not type-set. This allows resources to be used in multiple parts of the pipeline. Data from one shader can be used in another for on-the-fly updates, regardless of the format type of the intermediate data.

Examples of DX10 Visual Effects





Figure 8. DX9 Screenshot

Figure 9. DX10 Screenshot

(Source: Flight Simulator X game)

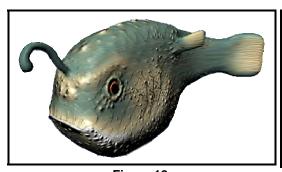




Figure 10. DX9 Screenshot

Figure 11. DX10 Screenshot

(Source: Crysis game)



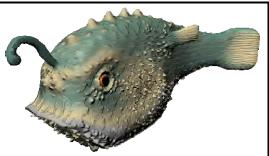


Figure 12.

Figure 13.

DX9 (normal mapping)

(Source: "DirectX 10: The Next Generation in Gaming" – http://windowsvistablog.com)

S3 Graphics DirectX 10 Architecture for Chrome 400 Series Processors

WP016-A.0 7/21/2007 Page 17 of 19

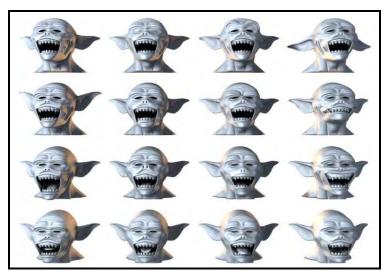


Figure 14. DX10 Morphing using the geometry shader and stream output (Source: Microsoft MSDN Direct3D 10 Samples, http://msdn2.microsoft.com)



Figure 15. DX10 Alpha to Coverage (Source: Microsoft "Intro to Direct3D 10" presentation by Sam Glassenberg)



Figure 16. DX10 Instancing
(Source: Microsoft MSDN Direct3D 10 Samples, http://msdn2.microsoft.com)

S3 Graphics DirectX 10 Architecture for Chrome 400 Series Processors WP016-A.0 7/21/2007 Page 18 of 19

Conclusion

The introduction of DirectX 10 brings an inflection point to the graphics market where the rendering capabilities of advanced hardware and the creativity of software developers can now bring real-life 3D graphics to our daily lives. Features, which once were found only in luxury high-end graphics products costing several thousands of dollars, can now be achieved using the new DirectX 10 companion GPUs of the S3 Graphics Chrome 400 Series product line.

S3 Graphics Chrome 400 Series graphics processors are technological marvels with their multiple programmable DX10/SM4.0 execution units which provide a unified shader architecture. With support for all the new DX10 features such as stream processing, geometry shaders, and HLSL 10 programming, S3 Graphics continues its position as an industry leader in visual computing for current and future generations of the DirectX 3D API.

DICTIONARY OF COMPUTING

FOURTH EDITION

S.M.H. Collin

PETER COLLIN PUBLISHING

ATI 2088 LG v. ATI IPR2015-00326 Fourth edition published 2002 Third edition published 1998 Second edition published 1994 First published in Great Britain 1988

Published by Peter Collin Publishing, an imprint of Bloomsbury Publishing Plc 38 Soho Square, London W1D 3HB © Copyright S.M.H. Collin, 1988, 1994, 1998, 2002

All rights reserved. No part of this publication may be reproduced in any form or by any means without the permission of the publishers.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

ISBN 1-901659-46-1

Text computer typeset by PCP
Printed and bound in Italy by Legoprint
Cover artwork by Gary Weston

vectored interrupt noun interrupt signal which directs the processor to a routine at a particular address

QUOTE: the great advantage of the vector-scan display is that it requires little memory to store a picture

Electronics & Power

374

Veitch diagram noun graphical representation of a truth table

velocity noun speed; the disk drive motor spins at a constant velocity

velocity (of sound) noun speed of sound which is equal to 331 metres per second through air; the speed of sound varies in different materials

vendor noun person who manufactures or sells or supplies hardware or software products; vendor independent = hardware or software that will work with hardware and software manufactured by other vendors; opposite is PROPRIETARY; vendor-independent messaging = see VIM vendor-independent messaging see VIM

Venn diagram graphical noun representation of the relationships between the states in a system or circuit

verification noun checking that data has been keyboarded correctly or that data transferred from one medium to another has been transferred correctly; keystroke verification = check made on each key pressed to make sure it is valid for a particular application; verification and validation (V & V) = testing a system to check that it is functioning correctly and that it is suitable for the tasks intended

verifier noun special device for verifying input data

verify verb to check that data recorded or entered is correct

Veronica tool that works with Gopher to help a user find information or files on the Internet

version noun copy or program or statement which is slightly different from others; the latest version of the software includes an improved graphics routine; version control = utility software that allows several programmers to work on a source file and monitors the changes that have been made by each programmer; version

number = number of the version of a product

vertex noun point in space defined by the three coordinates x, y, and z

vertical adjective at right angles to the application vertical horizontal: application software that has been designed for a specific use, rather than for general use; your new software to manage a florist's is a good vertical application; vertical blanking interval = see RASTER; vertical format unit (VFU) = part of the control system of a printer which governs the vertical format of the document to be printed (such as vertical spacing, page length); vertical interval time code = see VITC; vertical justification = adjustment of the spacing between lines of text to fit a section of text into a page; vertical parity check = error detection test in which the bits of a word are added and compared with a correct total; vertical portal (VORTAL) = website that contains information for just one particular industry or interest group; vertical redundancy check (VRC) = (odd) parity check on each character of a block received, to detect any errors; vertical scan frequency = number of times a picture beam in a monitor moves from the last line back up to the first; vertical scrolling = displayed text which moves up or down the computer screen one line at a time; vertical sync signal = (in a video signal) signal which indicates the end of the last trace at the bottom of the display; vertical tab = number of lines that should be skipped before printing starts again

vertically adverb from top to bottom or going up and down at right angles to the horizontal; the page has been justified vertically

very large scale integration (VLSI) noun integrated circuit with 10,000 to 100,000 components

VIDEO ELECTRONICS VESA = STANDARDS ASSOCIATION: VESA local bus or VL-bus = (in an IBM PC) standard defined by VESA which allows up to three special expansion slots that provide direct, bus-master control of the central processor and allow very high speed data transfers between main memory and the expansion card without using the processor:

A 195 mW/152 mW Mobile Multimedia SoC With Fully Programmable 3-D Graphics and MPEG4/H.264/JPEG

Jeong-Ho Woo, Student Member, IEEE, Ju-Ho Sohn, Associate Member, IEEE, Hyejung Kim, Student Member, IEEE, and Hoi-Jun Yoo, Fellow, IEEE

Abstract—In this paper, we present a low power multimedia SoC with fully programmable 3-D graphics, MPEG4 codec, H.264 decoder, and JPEG codec for mobile devices. The mobile unified shader in 3-D graphics engine provides fully programmable 3-D graphics pipeline with 35% area and 28% power reduction. Low power lighting engine which employs logarithmic number datapath and the specialized lighting instruction enable 9.1 Mvertices/s vertex fill rate, which is 2.5 times improvement compared with previous works including transformations and OpenGL lighting. The SoC consumes less than 152 mW for video applications and less than 195 mW for 3-D graphics applications. The mobile unified shader and merged JPEG/MPEG4 codec reduce the silicon area and the SoC consumes 6.4 mm \times 6.4 mm in 0.13 μ m CMOS logic process.

Index Terms—Low power design, mobile multimedia SoC, mobile unified shader, programmable 3-D graphics.

I. INTRODUCTION

RECENTLY, multiple multimedia applications are merged into the mobile devices to be a personal multimedia terminal [2]. The digital camera and real-time audio playback are widely incorporated and recently even digital multimedia broadcasting (DMB) and real-time 3-D graphics are employed for mobile entertainments. The real-time 3-D graphics has been used for various applications such as 3-D games or 3-D user-interfaces and recently portable navigation devices (PND) try to employ the real-time 3-D graphics for 3-D map displaying and 3-D navigation services.

In mobile devices, since users often hold the small screens closer to their eyes, the average eye-to-pixel angle is larger than that of a PC [3]. Therefore, every pixel in mobile applications should be drawn with realistic 3-D graphics effects, which can be achieved by a fully programmable 3-D graphics. In PC graphics, dedicated vertex shader and pixel shader carry out the fully programmable 3-D graphics and they make realistic 3-D images [9]. But, that PC graphics architecture cannot be migrated into the mobile devices due to its silicon area and power consumption.

Manuscript received October 29, 2007; revised May 5, 2008. Current version published September $10,\,2008.$

J.-H. Sohn is with LG Electronics Company, Ltd., Seoul 137-724, Korea (e-mail: sohnjuho@lge.com).

Digital Object Identifier 10.1109/JSSC.2008.2001911

Since the mobile devices restrict silicon area and power consumption, various multimedia applications should be implemented with small area and low power consumption. Although there have been many publications on mobile multimedia solutions [2]–[8], these chips did not integrate the full multimedia functions such as digital camera, video, audio and real-time 3-D graphics on a single die due to its huge gate counts and design complexity. Moreover, they could not provide a fully programmable 3-D graphics pipeline, which is required for realistic 3-D graphics effects compatible with OpenGL|ES-2.0.

In this work [1], a low power multimedia SoC with full integration of a fully programmable 3-D graphics and MPEG4, H.264 and JPEG processing is presented for mobile devices.

For the purpose to achieve low power, small area, and high performance, the programmable 3-D graphics engine with unique unified shader architecture [10] is employed. Its mobile unified shader is power optimized single shader for mobile devices in contrast to that of console device which has multiple general purpose unified shaders. The lighting engine and specialized lighting instruction are adopted for low power and high performance.

This paper consists of six sections. The system architecture and video engine will be discussed in Section II, and the programmable 3-D graphics engine and details of mobile unified shader will follow in Section III and Section IV, respectively. The chip implementation of the SoC and performance comparison will be described in Section V, and finally the conclusion of our work will be made in Section VI.

II. SYSTEM ARCHITECTURE AND VIDEO ENGINE

Fig. 1(a) shows a block diagram of the developed SoC. It integrates a 3-D graphics engine dedicated for acceleration of the fully programmable 3-D graphics pipeline, the ARM9 RISC processor, video engine, display engine and other peripheral IPs. Since most of the current mobile multimedia SoCs employ the AMBA bus so that the IPs are connected to the single layer AMBA bus. The video engine is employed to support video application such as DMB or digital camera. It is dedicated to MPEG encoding/decoding, H.264 decoding and JPEG image processing using dedicated hardwired blocks. Since both the JPEG codec and the MPEG codec use DCT, IDCT, VLC and VLD units, the JPEG and MPEG codec shares those functional blocks to reduce silicon area and power consumption. The fully hardwired H.264 decoder consists of a content-addressable variable-length decoder (CAVLD), an inverse-transform, a motion compensator and de-blocking filter. The video engine

0018-9200/\$25.00 © 2008 IEEE

ATI 2089 LG v. ATI IPR2015-00326

J.-H. Woo, H. Kim, and H.-J. Yoo are with the Division of Electrical Engineering, Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail: denber@eeinfo.kaist.ac.kr).

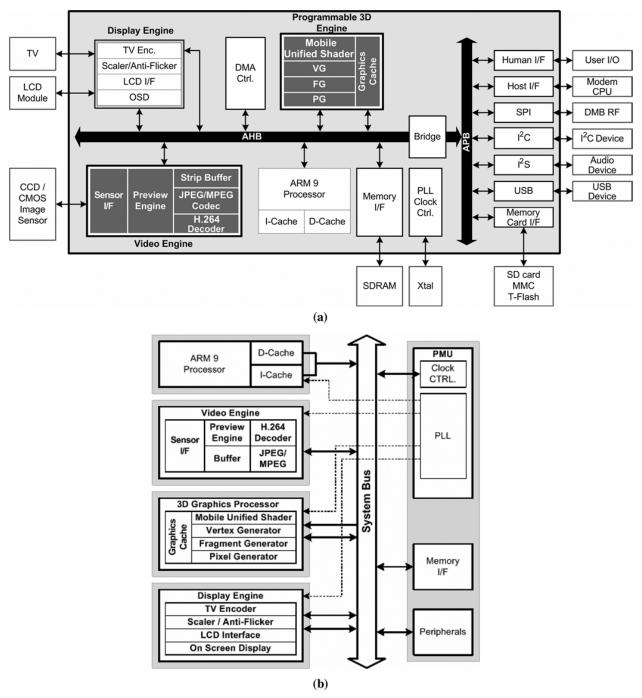


Fig. 1. Mobile multimedia SoC. (a) System architecture. (b) Power and clock domains.

supports up to 3M pixels image sensors, MPEG4 @ simple profile Lv.0 \sim 3 encoding/decoding and H.264 baseline Lv.0 \sim 3 decoding of 30 fps for CIF image resolution. During video operation, it consumes less than 152 mW at 1.2 V supply voltage and 48 MHz operating frequency.

For low power consumption, the developed SoC adopts block level clock gating and power gating. The SoC consists of five voltage islands and five clock domains as shown in Fig. 1(b). According to the operating mode, the power sources and clocks

of the functional blocks are selectively turned off. In the sleep mode, all IPs except wake-up logics are turned off and the SoC consumes less than 150 μ W. In normal operation mode, the RISC processor controls whether the IPs are turned off or not.

Since the IPs shares bus bandwidth, the bandwidth occupation is one of the important design issues in mobile multimedia SoC. During the 3-D graphics applications, the ARM processor computes application programs and transfer graphics processor (GPU) commands and primitive vertex indexes to the GPU

3D Graphics Engine Matrix / Quaternion GPU Register Vector Generator 128b Matrix Coefficient **Mobile Unified Shader** GPU Code SIMD Controller Memory Register File Cache SIMD Datapath Lighting Texture Engine Engine 256b 768b 1088b Vertex Gen. Pixel Gen. Fragment Gen. Clipping Alpha Blend Triangle Pespective Setup Division Frame Buf. Op. Pixel View-Port Mapping Interpolation Rus Controller Master Interface Slave Interface System Bus

Fig. 2. Programmable 3-D graphics processor.

through the system bus. Using the GPU commands, the GPU performs geometry operations and rendering operations. For geometry and rendering operations, the GPU fetches primitive vertex attributes, depth, color, and texture from the system memory. Those graphics data are transferred through the system bus and they consumes about 700 MBs/s if the GPU computes the graphics data at the speed of 10 Mvertices/s-296 MB/s for primitive vertices and 400 MB/s for rendering. Although the AMBA bus has 400 MB/s bus bandwidth, it can provide less than 120 MB/s bus bandwidth to GPU because the ARM processor and other peripherals operate simultaneously with the GPU. To reduce the data transaction of the GPU, the GPU employs vertex cache [4] for primitive vertex, texture cache with 1:4 texture compression for texture, and 2-D direct mapped cache [3] for color and depth. With those techniques, the required data bandwidth is reduced to 116.4 MB/s for drawing 10 Mvertices/s.

III. PROGRAMMABLE 3-D GRAPHICS ENGINE

A. Internal Architecture

The programmable 3-D graphics engine consists of Mobile Unified Shader, Vertex Generator, Fragment Generator, Pixel Generator, Matrix/Quaternion-Vector Generator and graphics caches as shown in Fig. 2.

The mobile unified shader is designed to perform both programmable vertex operation and programmable pixel operation, which are fully compatible with the mobile 3-D graphics API – OpenGL|ES2.0. Since the redundant functional blocks of the vertex shader and the pixel shader into a single hardware, the mobile unified shader reduces silicon area and power consumption. The SIMD datapath and special functional unit

TABLE I MATRIX GENERATOR COMMAND SET

Commands	Description
MG_Clear	Set Identify Matrix
MG_SET	Set Matrix With User Inputs
MG_SCALE	Scale a Matrix
MG_TRANSLATE	Translate a Matrix
MG_ROTATE	Rotate a Matrix
MG_TRANSPOSE	Transpose a Matrix
MG_INVERT	Inverse a Matrix
MG_Determinant	Get Determinant of Matrix
MG_MMUL	Matrix Multiplication

(SFU) of the mobile unified shader are responsible for the computational works and the texture engine [3] is responsible for texture address generation, texture fetch, and texture filtering. The vertex generator, fragment generator, and pixel generator are responsible for the rest of the 3-D graphics pipelines, clipping, triangle setup, rasterization, and blending, respectively. To reduce power consumption, the fragment generator employs low power SlimShader architecture [3].

During the per-vertex operation, the mobile unified shader uses several floating-point matrices related to rotation, transportation, scale, and projection and those matrices are generated by the embedded RISC processor. Since the embedded RISC processor has only integer datapaths, the floating point computations are done by complex emulation equations. Therefore, thousands of cycles are consumed for matrix operations and this computation cycles limit the 3-D graphics performance. In order to accelerate those matrix operations, the mobile unified shader employs the matrix generator. The matrix generator consists of 6 floating-point multipliers, 6 floating-point adders, a floating-point divider, a floating-point square-root block and a floating-point trigonometric function block and it is controlled by the RISC processor using the dedicated matrix generator instructions as shown in Table I. With those matrix instructions, the matrix generator calculates transformation and projection matrices by matrix basis while embedded RISC processor computes by components basis. In addition, since the matrix generator and the mobile unified shader have independent commands and datapaths respectively, user can use the matrix generator and the mobile unified shader at the same time. For example, when the shader program contains complex matrix operations, users can use matrix generator instead of shader datapath to simplify the shader operations.

B. Pixel-Vertex Multi Threading

Fig. 3(a) shows a data flow diagram of the programmable 3-D graphics pipeline. In conventional architecture, the primitive vertices are computed in the vertex shader for per-vertex operations such as transformation and lighting. After per-vertex operations, vertex generator and fragment generator perform clipping and rasterization and they generate interpolated pixels (fragments). After that, the fragments are modified in the pixel shader using per-pixel effects and blending operations generate final pixel data. In contrast with conventional architecture, the

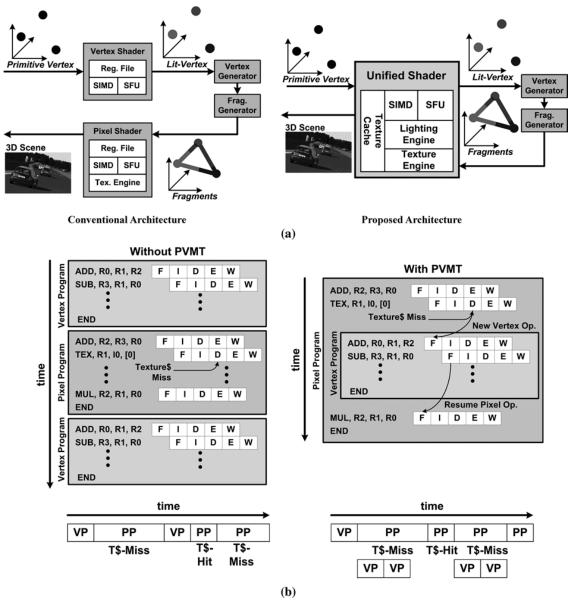


Fig. 3. Pixel-vertex multi-threading. (a) Data flow of the programmable 3-D graphics pipeline. (b) Pixel-vertex multi-threading.

mobile unified shader is responsible for both per-vertex operations and per-pixel operations in a single hardware. Therefore, the graphics data traverse the mobile unified shader twice in a single 3-D graphics pipeline as shown in Fig. 3(a) and thus, the 3-D graphics performance is limited less than the half of the its peak performance. Moreover, texture cache miss wastes a few tens of cycles until the cache is filled up and it degrades the graphics performance is degraded further.

To improve the 3-D graphics performance, the 3-D graphics processor adopts a Pixel-Vertex-Multi-Threading (PVMT), which utilizes datapaths of the mobile unified shader in parallel. Since the texture engine performs texture fetching and filtering independent with SIMD datapath and SFU, those datapaths are idle during the texture operations. Therefore, the PVMT enables SIMD datapath and SFU to compute per-vertex

operations during the texture cache miss as shown in Fig. 3(b). When the texture cache miss occurs during per-pixel operations and vertex buffer contains vertices to be computed, PVMT issues next vertices and SIMD datapath and SFU perform per-vertex operations. If the texture cache filling is finished during the per-vertex operations, the mobile unified shader moves back to per-pixel operation and finalize the remaining pixel operations. Otherwise, the PVMT issues next vertices and performs per-vertex operations continuously.

Since the PVMT uses wasted cycles by the texture cache miss, the efficiency of the PVMT depends on the application characteristics such as vertex/pixel ratio or texture cache miss rate. In mobile 3-D graphics, the texture cache miss is occurred with about 10% probability in average and it consumes as short as 64 cycles, or 148 cycles at the most in the developed mobile

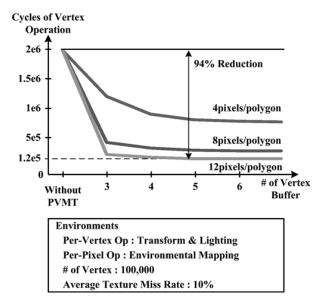


Fig. 4. Efficiency of pixel-vertex multi-threading.

multimedia SoC. Since the per-vertex operations such as transform and lighting (TnL) consume about 20 cycles in the test contents, more than three vertices could be computed during the texture cache miss if the vertex buffers are enough. Fig. 4 shows the efficiency of the PVMT versus the number of vertex buffers and the vertex/pixel ratio. While the effects of the PVMT vary from the vertex/pixel ratio, the PVMT reduces the cycles of the vertex operation as short as 60%, or 94% at the most. Although the PVMT reduces cycles of the vertex operations, the effect of the PVMT is bounded when the vertex buffer has five entries because the PVMT uses wasted cycles by the texture cache miss and the cycle counts are limited around 100 cycles in average. Therefore, the developed 3-D graphics processor has 5 entries vertex buffer for trade-off between hardware cost and the performance. By employing PVMT with five-entries vertex buffer, about 90% of the vertex operations are interleaved into the pixel operations and we can remove the cycle time of the vertex operations from the graphics pipeline.

IV. MOBILE UNIFIED SHADER

A. Internal Architecture

The mobile unified shader is a SIMD processor for fully programmable 3-D graphics. It consists of 128 bits, 4 × 32 bit SIMD datapath, SFU, texture engine, logarithmic lighting engine, dedicated register file and control logic as shown in Fig. 5. The SIMD datapath is responsible to vector arithmetic operations such as Addition (ADD), Multiplication (MUL) and inner product (DOT). The SFU is dedicated to special functional scalar operations such as Logarithm (LOG), Exponent (EXP), Reciprocal (RCP) and Reciprocal-square-root (RSQ). And the texture engine and the lighting engine perform texture related operations and lighting operations, respectively.

Since the LOG, EXP, RCP and RSQ operations are difficult to compute in ordinary number datapaths, the SFU employs the logarithmic number system (LNS) [11]. In the LNS, since the

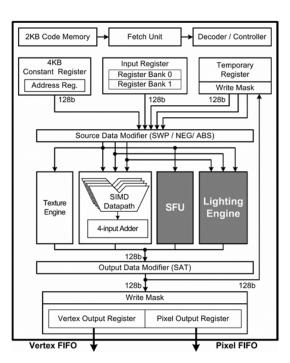


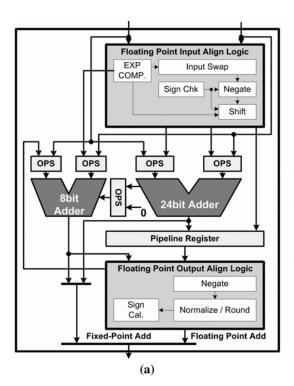
Fig. 5. Mobile unified shader architecture.

ordinary number data should be converted into the LNS and the results should be converted into ordinary number, one of the critical issues is the data conversion error. Therefore, we employed more precise LOG and EXP converts, which use 16 piecewise-linear-regions. By employing precise LNS datapath, the SFU computes the LOG, EXP, RCP and RSQ in only two cycles keeping the error bound much below the standard graphics API, OpenGL|ES-2.0. Since the vertex shading is performed in IEEE-754 floating point number system and pixel shading is performed in 32 bit fixed-point number system, the SFU is designed to handle both number systems in a single hardware.

For streaming graphics processing, the mobile unified shader contains multiple register files—input registers, output registers, constant register and temporary SIMD registers. The input register is used to hold the vertex attributes such as position and normal vector and pixel attributes such as position, color and texture coordinate. In order to reduce data fetch time, the input register consists of two register banks for double buffering. The constant register stores the coefficients for the 3-D graphics operations. The temporal register is used to store temporary results during vertex program and pixel program execution. The modified vertex and modified pixel information are transformed into output register.

B. Unified SIMD Datapaths

The SIMD datapath is responsible for vector and matrix arithmetic operations such as Addition (ADD), Multiplication (MUL), and Inner Product (DOT). Since, in mobile 3-D graphics API-OpenGL|ES, per-vertex operations use IEEE-754 floating-point number system for wide dynamic range and per-pixel operations use fixed-point number system for high throughput, the conventional vertex shader has floating-point



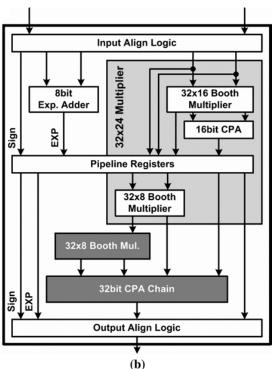


Fig. 6. Unified SIMD datapath. (a) Unified adder. (b) Unified multiplier.

SIMD datapath and pixel shader has fixed-point SIMD datapath. To compute both operations in a single mobile unified shader, we employed the unified datapaths, adder and multiplier, which computes IEEE-754 floating-point numbers and fixed-point numbers in a single hardware.

Since the fixed-point adder requires 32 bit addition and the floating point adder requires 8 bit addition for exponent and 24 bit addition for mantissa, the 32 bit adder can be shared for both number systems. Therefore, the unified adder has configurable 32 bit adder, which consists of a 24 bit adder and 8 bit adder as shown in Fig. 6(a). The configurable 32 bit adder is configured by operands selection and it computes floating-point addition with 2 cycle latency and 1 cycle throughput, and fixed-point addition in a single cycle. The unified multiplier consists of common 32 bit \times 24 bit multiplier, optional 32 bit \times 8 bit multiplier, and 8 bit adder for floating-point exponent as shown in Fig. 6(b) because the floating point multiplication includes 24 bit by 24 bit multiplication for mantissa and fixed point multiplication includes 32 bit by 32 bit multiplication. The final 32 \times 8 multiplier is conditionally enabled and the CPA chain selects the input between 24 bit result and 32 bit result. The unified multiplier calculates both floating-point MUL and floating-point MUL with 2 cycle latency and 1 cycle throughput. By sharing the common functional blocks of floating point datapath and fixed point datapath, the unified SIMD datapath reduces silicon area by 47% and power consumption by 42% compared with separated floating-point and fixed-point SIMD datapaths.

C. Low Power Lighting Engine

The lighting equation is the most complex operation during the vertex operation due to the power (POW) operation of specular lighting. To accelerate the lighting equation with low power consumption, the low power lighting engine is employed. The OpenGL lighting equation which includes an ambient light, a diffuse light and a specular light is described in (1):

$$Color_{ORD} = C_{amb} + \{ (N' \bullet L_{dir}) \times C_{diff} \} + \{ (N' \bullet H)^{C_{pow}} \times C_{spec} \}. \quad (1)$$

In conventional implementation [9], the lighting equation is performed sequential fashion as shown in Fig. 7(a). At first, the LIT instruction computes lighting coefficients. Then multiplications and additions are performed. During lighting computation, the data dependency wastes several cycles and it degrades graphics performance. However, as shown in (2), the lighting equation consists of three independent components and it can be computed in parallel:

$$Color_{LNS} = C_{amb} + \{ (N' \bullet L_{dir}) \times C_{diff} \} + 2 \{ log_2(N' \bullet H) \times C_{pow} + log_2 C_{spec} \}. \quad (2)$$

To improve lighting computation, the lighting engine and specialized lighting instruction are designed to compute three components simultaneously. Since the POW operation can be converted into a multiplication as shown in (2), the lighting engine employs the logarithmic number datapaths [11] for specular lighting. Therefore, the lighting engine consists of ordinary number datapaths for ambient and diffuser lighting computation and logarithmic number datapaths to accelerate specular lighting computation as shown in Fig. 7(b). To utilize the lighting engine efficiently by graphics APIs, the specialized lighting instruction, TLT, is proposed. The TLT instruction

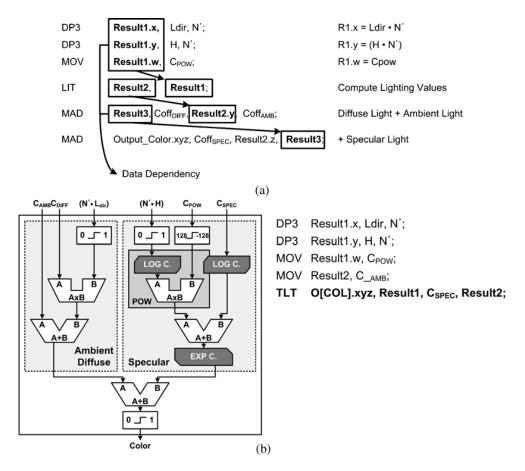


Fig. 7. Low power lighting engine. (a) Conventional lighting computation. (b) Proposed lighting engine.

combines the coefficient calculation and the multiplication of coefficients and materials together as shown in Fig. 7(b). The lighting engine with TLT instruction computes the lighting equation without data dependency so that, the lighting engine with TLT instruction generates lit vertex in every two cycles. For compatibility with OpenGL|ES interface, the lighting engine supports both the conventional LIT instruction and TLT instruction. In the conventional lighting operation, the lighting engine accelerates the POW operations and it computes only the lighting coefficients, which are used for complex 3-D graphics effects such as user-defined lighting. Otherwise, the TLT instruction is used to accelerate the common lighting environments, which is widely used in commercial mobile 3-D contents. By adopting logarithmic LE and TLT instruction, the mobile unified shader generates a lit-vertex in every two cycles and it achieves 9.1 Mverticies/s, which is 2.5 times higher performance compared with previous implementation [3].

D. Micro-Level Power Management

For low power consumption of the mobile unified shader, it implements instruction-level power management. To activate the 3-D graphics processor, the ARM processor must drive activation signal to 3-D graphics processor and the 3-D GPU controller drive enable signal to the unified shader only when the

current shader instruction is valid. Using this enable signal, the clock signals of the SIMD register files can be gated off when the write operations of the register files are not required. The read operations of the register files are still possible in the clock-off state. Like the same way, the enable signal also reduces the power dissipation of SIMD arithmetic units by eliminating the unnecessary signal transition. Therefore, the power consumption of the mobile unified shader is controlled on an instruction level. In the mobile unified shader, since the SIMD register files and datapath consumes about 80% of power, about 85% activation ratio in full 3-D graphics pipeline achieves up to 12% power reduction of the mobile unified shader. For power management of the pixel-operations, it implements pixel-level clock gating. The pixel operation includes fragment generator for triangle setup and rasterization, mobile unified shader for per-pixel operations, and pixel generator for blending operations. To reduce power consumption, the fragment generator allows clock gating, which uses depth-comparison results generated in early stage of rendering pipeline. If a new pixel to be drawn is already covered by the pixels near from the viewpoint, the remaining operations does not need to process further. To use this property, per-pixel operations of the mobile unified shader is killed and the clock signal of the pixel generator is gated-off to prevent unnecessary operations.

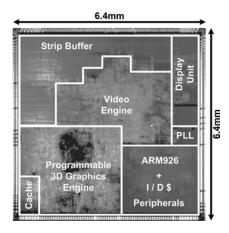


Fig. 8. Die microphoto.

TABLE II FEATURE SUMMARY

Process Technology		0.13μm 7M CMOS		
Powe	er Supply	1.2V		
Transis	stor Counts	18.6M Transistors (including 128KB SRAM)		
Die Size		6.4mm x 6.4mm		
Operating Frequency		100MHz		
Power Consumption		152mW @ Video Processing 195mW @ Full 3D Graphics Processing		
Pa	ackage	144pin CABGA		
	Image Processing	Up to 3M pixels Camera Resolution Support Real-Time Baseline JPEG Encoding / Decoding		
Multimedia Performance	Video Processing	MPEG : CIF 30 fps Codec @ SP Lv.0~3 H.264 : CIF 30 fps Decode @ BL Lv.0~3		
	3D - Geometry	9.1Mvertices/s (Including Full OpenGL Lighting)		
	3D - Rendering	100Mpixels/s, 400Mtexels/s		
	Programmability	Programmable Vertex Shader Programmable Pixel Shader		
Graphics Functions	Standard API	OpenGL ES ver. 1.1 OpenGL ES ver. 2.0		
Screen Resolution		Up to 1024 x 1024 pixels		

V. IMPLEMENTATION RESULTS

The mobile multimedia SoC is fabricated using a 0.13 μ m seven-metal CMOS logic process. It contains 18.6 M transistors including 128 KB SRAM in 6.4 \times 6.4 mm². Fig. 8 shows the chip microphotograph and Table II summarizes chip features. By using this chip, realistic 3-D graphics effects can be processed with 9.1 Mvertices/s peak graphics performance and 30 fps MPEG4 encoding/decoding and 30 fps H.264 decoding can be processed in the mobile devices. The 3-D graphics images with realistic graphics effects are successfully demonstrated by the fabricated chip on the system evaluation board as shown in Fig. 9. For a demo content which uses both the per-vertex operations such as transformation and lighting and the per-pixel operations such as environmental mapping, the developed SoC continuously draws the 3-D images at the speed of 7.4 Mpixels/s.

With the mobile unified shader, the developed SoC can provide fully programmable 3-D graphics pipeline, per-vertex



Fig. 9. System evaluation board.

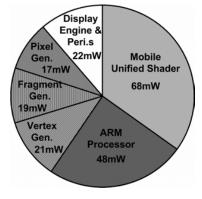


Fig. 10. Power dissipation of the 3-D graphics applications.

operations and per-pixel operations, and thus, it can completely generate realistic 3-D graphics effects such as environmental mapping or per-pixel lighting. The proposed mobile unified shader reduces silicon area of the 3-D graphics processor by 35% with the help of the unified SIMD datapaths, logarithmic datapath. The logarithmic datapaths, low power register file, and micro-level (instruction-level and pixel-level) clock gating reduces the power consumption by 28%. The developed SoC consumes 195 mW in continuous calculation of 9.1 Mvertices/s full 3-D graphics pipeline including programmable per-vertex operations and programmable per-pixel operations at 100 MHz operating frequency and 1.2 V supply voltage and it consumes 152 mW in continuous calculation of CIF 30 fps H.264 decoding including audio playback at 48 MHz operating frequency and 1.2 V supply voltage. The power dissipations of the functional blocks are summarized in Fig. 10.

Fig. 11 shows the 3-D graphics performance comparison with that of the previous implementations [3], [5], [12]. The 3-D graphics performance is measured including transformation and OpenGL lighting with one ambient light, one diffuse light, and one specular light for geometry and texture blending for rendering. The developed SoC provides 9.1 Mvertices/s fully programmable 3-D graphics including transformation and lighting and, with the help of the lighting engine and specialized lighting instruction, it improves 2.1 times vertex fill rate compared with

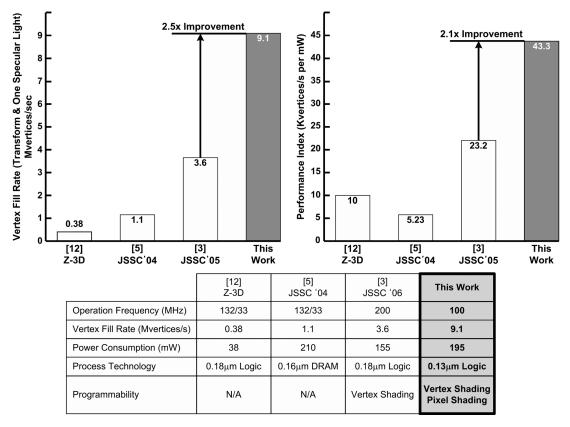


Fig. 11. 3-D graphics performance comparison.

previous implementation [3]. The 3-D graphics performance in the mobile devices cannot be compared directly in terms of processing speed such as vertex fill rate because the power consumption must be taken into account as well [5]. Based on the performance index of the mobile 3-D graphics [3], which is the vertex fill rate normalized by the power consumption, the developed SoC shows 46.67 Kvertices/s/mW, which is about 2 times improvement over the previous implementation [3].

VI. CONCLUSION

A low power multimedia SoC is designed for mobile devices. It integrates fully programmable 3-D graphics engine, MPEG4/JPEG codec and H.264 decoder in a single chip. The mobile unified shader provides a fully programmable 3-D graphics with 35% area reduction and 28% power reduction. Low power lighting engine and the specialized lighting instruction achieves 9.1 MVertices/s vertex fill rate which is 2.5 times improvement over previous works including transformation and OpenGL lighting. The PVMT improves graphics performance by interleaving the vertex operations into pixel operations and as a result, up to 94% vertex operations are hided into the pixel operations. The SoC consumes less than 195 mW at 1.2 V supply voltage and 100 MHz operating frequency for 3-D graphics and less than 152 mW at 1.2 V supply voltage and 48 MHz operating frequency for video operations. With the help of the mobile unified shader and merged JPEG/MPEG4 codec engine, the SoC reduces silicon area and it is implemented in 6.4 mm \times 6.4 mm using 0.13 μ m CMOS logic process.

REFERENCES

- [1] J. H. Woo et al., "A 152mW mobile multimedia SoC with fully programmable 3D graphics and MPEG4/H.264/JPEG," in *IEEE Symp. VLSI Circuits*, 2007, pp. 220–221.
- [2] J. H. Sohn et al., "Low power 3D graphics processors for mobile terminals," *IEEE Commun. Mag.*, vol. 43, no. 12, pp. 90–99, Dec. 2005.
- [3] J. H. Sohn et al., "A 155-mW 50-Mvertices/s graphics processor with fixed-point programmable vertex shader for mobile applications," *IEEE J. Solid-State Circuits*, vol. 41, no. 5, pp. 1081–1091, May 2006.
- [4] B. G. Nam et al., "A low-power unified arithmetic unit for programmable handheld 3-D graphics systems," IEEE J. Solid-State Circuits, vol. 42, no. 8, pp. 1767–1778, Aug. 2007.
- [5] R. Woo et al., "A 210 mW graphics lsi implementing full 3-D pipeline with 264 Mtexels/s texturing for mobile multimedia applications," in IEEE ISSCC Dig. Tech. Papers, 2003, pp. 44–45.
- [6] T. M. Liu et al., "A 125 μW, fully scalable MPEG-2 and H.264/AVC video decoder for mobile applications," in *IEEE ISSCC Dig. Tech. Papers*, 2006, pp. 402–403.
- [7] T. Hashimoto et al., "A 27-MHz/54-MHz 11-mW MPEG-4 video decoder LSI for mobile applications," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1574–1581, Nov. 2002.
- [8] M. Imai et al., "A 109.5 mW 1.2 V 600 Mtexel/s 3D graphics engine," in IEEE ISSCC Dig. Tech. Papers, 2004, pp. 332–333.
- [9] J. Montrym et al., "Geforce4," in ACM SIGGRAPH/Eurograph Graphics Hardware Workshop, 2002.
- [10] M. Doggett, "Xenos: XBOX360 GPU," in ACM Eurographics, 2005.
- [11] H. Kim et al., "A 231 MHz, 2.18 mW 32 bit logarithmic arithmetic unit for fixed-point 3-D graphics system," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2373–2381, Nov. 2006.
- [12] M. Kameyama et al., "3-D graphics LSI core for mobile phone "Z3-D"," in Graphics Hardware Workshop, 2003.



Jeong-Ho Woo (S'02) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2002 and 2004, respectively. He is currently working toward the Ph.D. degree in electrical engineering at KAIST.

His research includes design of low-power mobile multimedia system with 3-D computer graphics and its implementation. In particular, he is now working on a low-power programmable 3-D graphics processor for mobile devices.



Ju-Ho Sohn (A'07) received the B.S. (*summa cum laude*), M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001, 2003, and 2006, respectively. During the period in KAIST, he developed low-power programmable 3-D graphics processor integrated with vertex shader and DSP engine.

His research interests include low-power high-performance circuits and multimedia system design with specific interest in 3-D computer graphics and video

processing architecture. Currently, he is with LG Electronics, Seoul, Korea, designing a digital TV image enhancement chip.



Hyejung Kim (S'04) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2004 and 2006, respectively. She is currently working toward the Ph.D. degree in the same department at KAIST.

Her research interests include low-power arithmetic unit and microprocessor design for mobile and implantable system applications.



Hoi-Jun Yoo (M'95–SM'04–F'08) graduated from the Electronic Department of Seoul National University, Seoul, Korea, in 1983 and received the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1985 and 1988, respectively. His Ph.D. work concerned the fabrication process for GaAs vertical optoelectronic integrated circuits.

From 1988 to 1990, he was with Bell Communications Research, Red Bank, NJ, where he invented the

two-dimensional phase-locked VCSEL array, the front-surface-emitting laser, and the high-speed lateral HBT. In 1991, he became Manager of a DRAM design group at Hyundai Electronics and designed a family of fast-1 M DRAMs and 256 M synchronous DRAMs. In 1998 he joined the faculty of the Department of Electrical Engineering at KAIST and now is a full Professor. From 2001 to 2005, he was the Director of the System Integration and IP Authoring Research Center (SIPAC), funded by the Korean government to promote worldwide IP authoring and its SoC application. From 2003 to 2005, he was the full-time Advisor to the Minister of Korea Ministry of Information and Communication and National Project Manager for SoC and Computer. In 2007, he founded SDIA (System Design Innovation and Application Research Center) at KAIST to research and develop SoCs for intelligent robots, wearable computers and biosystems. His current interests are high-speed and low-power networks-on-chips, 3-D graphics, body area networks, biomedical devices and circuits, and memory circuits and systems. He is the author of the books DRAM Design (Hongleung, 1996; in Korean), High Performance DRAM (Sigma, 1999; in Korean), and chapters of Networks on Chips (Morgan Kaufmann, 2006).

Dr. Yoo received the Electronic Industrial Association of Korea Award for his contribution to DRAM technology in 1994, the Hynix Development Award in 1995, the Korea Semiconductor Industry Association Award in 2002, the Best Research of KAIST Award in 2007, the Design Award of 2001 ASP-DAC, and Outstanding Design Awards at 2005, 2006, and 2007 A-SSCC. He is a member of the executive committees of ISSCC, Symposium on VLSI, and A-SSCC. He is the TPC chair of A-SSCC 2008.



Technical Brief

NVIDIA GeForce® GTX 200 GPU Architectural Overview

Second-Generation Unified GPU Architecture for Visual Computing

ATI 2090 LG v. ATI IPR2015-00326

Table of Contents

Introduction4
GeForce GTX 200 Architectural Design Goals and Key Capabilities5
Architectural Design Goals 5
Gaming Beyond: Dynamic 3D Realism
Gaming Beyond: Extreme HD7
Gaming Beyond: SLI7
Beyond Gaming: High-Performance Visual Computing and Professional Computation8
GeForce GTX 200 GPU Architecture9
More Processor Cores9
Graphics Processing Architecture
Parallel Computing Architecture
SIMT Architecture
Greater Number of Threads in Flight
Larger Register File
Improved Dual Issue
Double Precision Support
Improved Texturing Performance
Higher Shader to Texture Ratio
ROP Improvements
1 GB Framebuffer
Geometry Shading and Stream Out
512-bit Memory Interface
Power Management Enhancements
Additional Pipeline and Architecture Enhancements
Summary
Appendix A: Retrospective21
Appendix B: Figure 1 References22

Figures

Figure 1: Realistic warrior from NVIDIA "Medusa" demo	6
Figure 2: Far Cry 2 – Extreme HD Dynamic Beauty! (Ubisoft)	7
Figure 3: Significant Speedup Using GPU	8
Figure 4: GeForce GTX 280 GPU Graphics Processing Architecture	10
Figure 5: GeForce GTX 280 GPU Parallel Computing Architecture	12
Figure 6: TPC (Thread Processing Cluster)	13
Figure 7: Local Register File 2× versus 1×	14
Figure 8: Geometry Shading Performance	17
	Tables
Table 1: Number of GPU Processing Cores	9
Table 2: GeForce 8800 GTX vs GeForce GTX 280	11
Table 3: Maximum Number of Threads	14
Table 4: Theoretical vs Measured Texture Filtering Rates	16

Introduction

In this technical brief we introduce NVIDIA's new GeForce® GTX 200 GPU family, the first GPUs to implement NVIDIA's second-generation unified graphics and computing architecture. The high-end, enthusiast-class GeForce GTX 280 GPU and performance-oriented GeForce GTX 260 GPU are the first members of the GeForce GTX 200 GPU family and deliver the ultimate visual computing and extreme high-definition (HD) gaming experience.

We'll begin by describing architectural design goals and key features, and then dive into the technical implementation of the GeForce GTX 200 GPUs. We assume you have a basic understanding of first-generation NVIDIA unified GPU architecture, including unified shader design, scalar processing cores, decoupled texture and math units, and other architectural features. If you are not well versed in NVIDIA unified GPU architecture, we suggest you first read the Technical Brief titled NVIDIA GeForce 8800 GPU Architecture Overview. You can also refer to Appendix A for a historical retrospective.

GeForce GTX 200 Architectural Design Goals and Key Capabilities

GeForce GTX 200 GPUs are massively multithreaded, many-core, visual computing processors that incorporate both a second-generation unified graphics architecture and an enhanced high-performance, parallel-computing architecture.

Two overarching themes drove GeForce GTX 200 architectural design and are represented by two key phrases: **"Beyond Gaming"** and **"Gaming Beyond."**

Beyond Gaming means the GPU has evolved beyond being used primarily for 3D games and driving standard PC display capabilities. More and more, GPUs are accelerating non-gaming, computationally-intensive applications for both professionals and consumers.

Gaming Beyond means that the GeForce GTX 200 GPUs enable amazing new gaming effects and dynamic realism, delivering much higher levels of scene and character detail, more natural character motion, and very accurate and convincing physics effects.

The GeForce GTX 200 GPUs are designed to be fully compliant with Microsoft DirectX 10 and Open GL 2.1.

Architectural Design Goals

NVIDIA engineers specified the following design goals for the GeForce GTX 200 GPUs:

- Design a processor with up to twice the performance of GeForce 8800
- Rebalance the architecture for future games that use more complex shaders and more memory
- ☐ Improve architectural efficiency per watt and per square millimeter
- ☐ Improve performance for DirectX 10 features such as geometry shading and stream out
- □ Provide significantly enhanced computation ability for highperformance CUDA™ applications and GPU physics
- Deliver improved power management capability, including a substantial reduction in idle power.

GeForce GTX 200 GPUs enable major new graphics and compute capabilities, providing the most realistic 3D graphics effects ever rendered by GPUs to date, while also providing nearly a teraflop of computational power.

May 2008 | TB-04044-001_v01

Gaming Beyond: Dynamic 3D Realism

While prior-generation GPUs could deliver real-time images that appeared true-to-life in many cases, frame rates could drop to unplayable levels in complex scenes with significant animation, numerous physical effects, and multiple characters. The combination of the sheer shader processing power of GeForce GTX 200 GPUs and NVIDIA's new PhysX™ technology facilitates many new high-end graphics effects including:

- ☐ Convincing facial and character animation
- ☐ Multiple ultra-high polygon characters in complex environments
- ☐ Advanced volumetric effects (smoke, fog, mist, etc.)
- ☐ Fluid and cloth simulation
- Fully simulated physical effects such as live debris, explosions, and fires.
- Physical weather effects such as accumulating snow and water, sand storms, soaking, drying, dampening, overheating, and freezing
- Better lighting for dramatic and spectacular effect, including ambient occlusion, global illumination, soft shadows, color bleeding, indirect lighting, and accurate reflections.



Figure 1: Realistic warrior from NVIDIA "Medusa" demo

6 May, 2008 | TB-04044-001_v01

Gaming Beyond: Extreme HD

GeForce GTX 200 GPUs provide 50-100% more performance over priorgeneration GPUs, permitting increased frame rates and higher visual quality settings at extreme resolutions, resulting in a truly cinematic gaming experience.

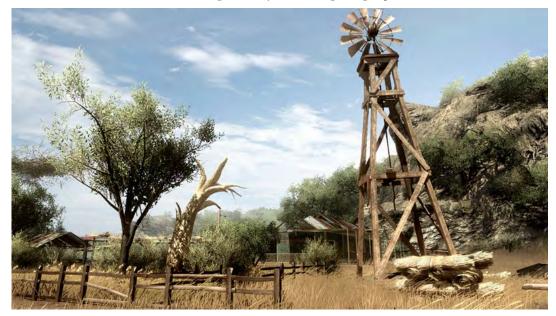


Figure 2: Far Cry 2 - Extreme HD Dynamic Beauty! (Ubisoft)

Support for the new DisplayPort interface allows resolutions beyond 2560×1600 , and 10-bit color support permits up to a billion different colors on screen (driver, display, and application support is also required). Note that prior-generation GPUs included internal 10-bit processing, but could only output 8-bit component colors (RGB). GeForce GTX 200 GPUs permit both 10-bit internal processing and 10-bit color output.

Gaming Beyond: SLI

NVIDIA's SLI® technology is the industry's leading multi-GPU technology, giving you an easy, low-cost, high-impact performance upgrade. PC gaming simply doesn't get any faster or more realistic than running GeForce GTX 200 GPU-based boards in SLI mode on the latest nForce® motherboards.

Two flavors of SLI are supported by the initial GeForce GTX 200 GPUs:

- □ Standard SLI (two GPU boards), which typically boosts supported game performance by 60-90% and permits higher quality settings
- 3-way SLI, which provides even higher frame rates and permits higher quality settings for the ultimate experience in PC gaming when connected to a high-end, high-resolution monitor.

GeForce GTX 200 GPUs process and display complex DirectX 10 and OpenGL game environments with amazing graphics effects and high frame rates at extreme, high-definition resolutions.

Beyond Gaming: High-Performance Visual Computing and Professional Computation

With the power of CUDA technology and the new CUDA runtime for Windows Vista, intensive computational tasks can be offloaded from the CPU to the GPU. GeForce GTX 200 GPUs can accelerate numerous rich-media and computationally-intensive applications such as video and audio transcoding, or running distributed computing applications like Folding@home in the background while surfing the web. Examples of GPU-enabled applications include the RapidHD video transcoding application from Elemental and various video and photo editing applications.

Many engineering, scientific, medical, and financial areas demand high-performance computational horsepower for numerous applications.

Figure 3 shows the amazing speedups that can be achieved by using a GPU instead of a CPU in a number of professional visual computing applications, in addition to mainstream video transcoding. Appendix B lists references and details for these applications.

Speedups Using GPU vs CPU

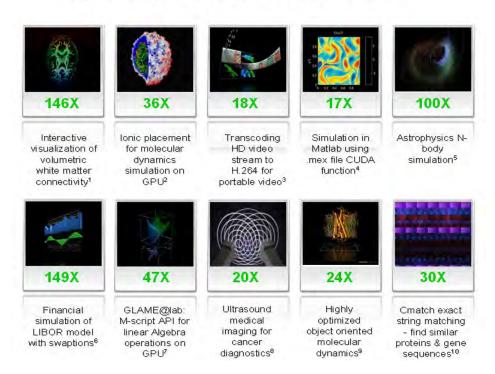


Figure 3: Significant Speedup Using GPU

With an understanding of the GeForce GTX 200 GPU design goals and key objectives, let's delve deeper into its internal architecture, looking at both the graphics and parallel processing capabilities.

GeForce GTX 200 GPU Architecture

GeForce GTX 200 GPUs are the first to implement NVIDIA's second-generation unified shader and compute architecture. The GeForce GTX 200 GPUs include significantly enhanced features and deliver, on average, 1.5× the performance of GeForce 8 or 9 Series GPUs.

Manufactured using TSMC's 65 nm fabrication process, GeForce GTX 200 GPUs include 1.4 billion transistors and are the largest, most powerful, and most complex GPU ever made. All GTX 200 GPUs are built to operate comfortably within the power and heat specifications of high-end PCs.

You may recall that the first-generation NVIDIA unified visual computing architecture in GeForce 8 and 9 Series GPUs was based on a Scalable Processor Array (SPA) framework. The second-generation architecture in GeForce GTX 200 GPUs is based on a reengineered, enhanced, and extended SPA architecture.

The SPA architecture consists of a number of TPCs, which stands for "Texture Processing Clusters" in graphics processing mode, and "Thread Processing Clusters" in parallel compute mode. Each TPC is in turn made up of a number of streaming multiprocessors (SMs), and each SM contains eight processor cores (also called streaming processors (SPs) or thread processors). Every SM also includes texture filtering processors used in graphics processing, but also useful for various filtering operations in compute mode, such as filtering images as they are zoomed in and out.

More Processor Cores

The new second-generation SPA architecture in the GeForce GTX 280 improves performance compared to the prior generation G80 and G92 designs on two levels. First, it increases the number of SMs per TPC from two to three. Second, it increases the maximum number of TPCs per chip from 8 to 10. The effect is multiplicative, resulting in 240 processor cores.

Chip	TPCs	SMs per TPC	SPs per SM	Total SPs
GeForce 8 & 9 Series	8	2	8	128
GeForce GTX 200 GPUs	10	3	8	240

Table 1: Number of GPU Processing Cores

Based on traditional processing core designs that can perform integer and floatingpoint math, memory operations, and logic operations, each processing core is a hardware-multithreaded processor with multiple pipeline stages that execute an instruction for each thread every clock.

Various types of threads exist, including pixel, vertex, geometry, and compute. For graphics processing, threads execute a shader program and many related threads often simultaneously execute the same shader program for greater efficiency.

All GeForce GTX 200 GPUs include a substantial portion of die area dedicated to processing, unlike CPUs where a majority of die area is dedicated to onboard cache memory. Rough estimates show 20% of the transistors of a CPU are dedicated to computation, compared to 80% of GPU transistors. GPU processing is centered on computation and throughput, where CPUs focus heavily on reducing latency and keeping their pipelines busy (high cache hit rates and efficient branch prediction).

Graphics Processing Architecture

As mentioned earlier, the GeForce GTX 200 GPUs include two different architectural personalities—graphics and computing. Figure 4 represents the GeForce 280 GTX in graphics mode. You can see the shader thread dispatch logic at the top, in addition to setup and raster units. The ten TPCs each include three SMs, and each SM has 24 processing cores for a total of 240 scalar processing cores. ROP (raster operations processors) and memory interface units are located at the bottom.

GeForce GTX 280 Graphics Processing Architecture

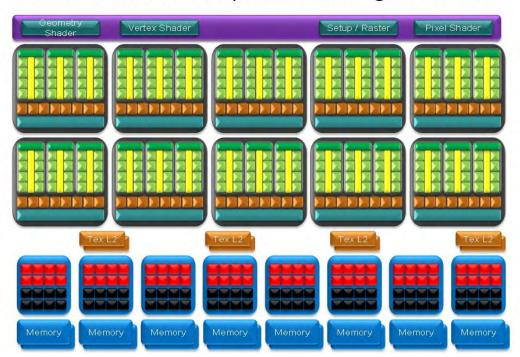


Figure 4: GeForce GTX 280 GPU Graphics Processing Architecture

Although not apparent in the above diagram, the architectural efficiency of the GeForce GTX 200 GPUs is substantially enhanced over the prior generation. We'll be discussing many areas that were improved in more detail, such as texture processing, geometry shading, dual issue, and stream out. In directed tests, GeForce GTX 200 GPUs can attain efficiencies closer to the theoretical performance limits than could prior generations.

Table 2 compares the GeForce 8800 GTX to the new GeForce GTX 280 GPU. You will notice sizable increases in a number of important measurable parameters.

Features	8800 GTX	GTX 280	% Increase
Cores	128	240	87.5 %
TEX	64t/clk	80t/clk	25 %
ROP Blend	12p/clk	32p/clk	167 %
Precision	fp32	fp64	
GFLOPs	518	933	80 %
FB Bandwidth	86 GB	142 GB	65 %
Texture Fill	37 GT/s	48 GT/s	29.7 %
ROP Blend	7 GBL/s	19 GBL/s	171 %
PCI Express	6.4 GB	12.8 GB	100 %
Video	VP1	VP2	

Table 2: GeForce 8800 GTX vs GeForce GTX 280

Parallel Computing Architecture

Figure 5 depicts a high-level view of the GeForce GTX 280 GPU parallel computing architecture. A hardware-based thread scheduler at the top manages scheduling threads across the TPCs. You'll also notice the compute mode includes texture caches and memory interface units. The texture caches are used to combine memory accesses for more efficient and higher bandwidth memory read/write operations. The elements indicated as "atomic" refer to the ability to perform atomic read-modify-write operations to memory. Atomic access provides granular access to memory locations and facilitates parallel reductions and parallel data structure management.

GeForce GTX 280 Parallel Computing Architecture



Figure 5: GeForce GTX 280 GPU Parallel Computing Architecture

A TPC in compute mode is represented in Figure 6 below. You can see local shared memory is included in each of the three SMs. Each processing core in an SM can share data with other processing cores in the SM via the shared memory, without having to read or write to or from an external memory subsystem. This contributes greatly to increased computational speed and efficiency for a variety of algorithms.

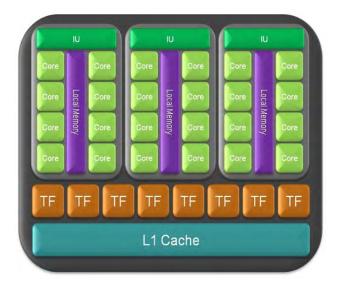


Figure 6: TPC (Thread Processing Cluster)

SIMT Architecture

NVIDIA's unified shading and compute architecture uses two different processing models. For execution across the TPCs, the architecture is MIMD (multiple instruction, multiple data). For execution across each SM, the architecture is SIMT (single instruction, multiple thread).

SIMT improves upon pure SIMD (single instruction, multiple data) designs in both performance and ease of programmability. Being scalar, SIMT has no set vector width and therefore performs at full speed irrespective of vector sizes.

In contrast, SIMD machines operate at a reduced capacity if the input is smaller than the MIMD or SIMD width. SIMT ensures the processing cores are fully utilized at all times.

From the programmer's perspective, SIMT also allows each thread to take on its own path. Since branching is handled by the hardware, there is no need to manually manage branching within the vector width.

Greater Number of Threads in Flight

GeForce GTX 200 GPUs support over thirty thousand threads in flight. Hardware thread scheduling ensures all processing cores attain nearly 100% utilization. The GPU architecture is latency-tolerant—if a particular thread is waiting for a memory access, the GPU can perform zero-cost hardware-based context switching and immediately switch to another thread to process.

The SIMT multithreaded instruction unit within an SM creates, manages, schedules, and executes threads in groups of 32 parallel threads called "warps." Up to 32 warps/SM are supported in GeForce GTX 200 GPUs, versus 24 warps/SM in GeForce 8 or 9 Series GPUs.

Chip	TPCs	SM per TPC	Threads per SM	Total Threads Per Chip
GeForce 8 & 9 Series	8	2	768	12,288
GeForce GTX 200 GPUs	10	3	1,024	30,720

Table 3: Maximum Number of Threads

Doing the math results in 32 x 32, or 1,024 maximum concurrent threads that can be managed by each SM. With 30 SMs in total, the GeForce GTX 280 supports up to 30,720 concurrent threads in hardware (versus 768 threads/SM \times 2 SMs/TPC \times 8 TPCs = 12,288 maximum concurrent threads in GeForce 8800 GTX).

Larger Register File

The local register file size has doubled per SM in GeForce GTX 200 GPUs compared to GeForce 8 & 9 Series GPUs. The older GPUs could run into situations with long shaders where registers would be exhausted, generating the need to swap to memory. A much larger register file permits larger and more complex shaders to be run on the GeForce GTX 200 GPUs faster and more efficiently. In terms of die size increase, the additional register file takes only a small fraction of SM die area.

Games are employing more and more complex shaders that require more register space. Figure 7 below highlights performance improvements 2× register file size in 3D Mark Vantage.

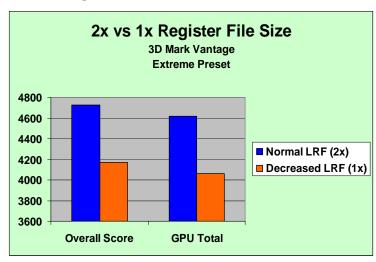


Figure 7: Local Register File 2× versus 1×

14 May, 2008 | TB-04044-001_v01

Improved Dual Issue

Special function units (SFUs) in the SMs compute transcendental math, attribute interpolation (interpreting pixel attributes from a primitive's vertex attributes), and perform floating-point MUL instructions. The individual streaming processing cores of GeForce GTX 200 GPUs can now perform near full-speed dual-issue of multiply-add operations (MADs) and MULs (3 flops/SP) by using the SP's MAD unit to perform a MUL and ADD per clock, and using the SFU to perform another MUL in the same clock. Optimized and directed tests can measure around 93-94% efficiency.

The entire GeForce GTX 200 GPU SPA delivers nearly one teraflop of peak, single-precision, IEEE 754, floating-point performance.

Double Precision Support

A very important new addition to the GeForce GTX 200 GPU architecture is double-precision, 64-bit floating point computation support. This benefits various high-end scientific, engineering, and financial computing applications or any computational task requiring very high accuracy of results. Each SM incorporates a double-precision 64-bit floating math unit, for a total of 30 double-precision 64-bit processing cores.

The double-precision unit performs a fused MAD, which is a high-precision implementation of a MAD instruction that is also fully IEEE 754R floating-point specification compliant. The overall double-precision performance of all 10 TPCs of a GeForce GTX 280 GPU is roughly equivalent to an eight-core Xeon CPU, yielding up to 78 gigaflops.

Improved Texturing Performance

The eight TPCs of the GeForce 8800 GTX allowed for 64 pixels per clock of texture filtering, 32 pixels per clock of texture addressing, 32 pixels per clock of 2× anisotropic bilinear filtering (8-bit integer), or 32-bilinear-filtered pixels per clock (8-bit integer or 16-bit floating point). Subsequent GeForce 8 and 9 Series GPUs balanced texture addressing and filtering.

☐ For example, the GeForce 9800 GTX can address and filter 64 pixels per clock, supporting 64-bilinear-filtered pixels per clock (8-bit integer) or 32-bilinear-filtered pixels per clock (16-bit floating point).

GeForce GTX 200 GPUs also provide balanced texture addressing and filtering and each of the 10 TPCs includes a dual-quad texture unit capable of addressing and filtering eight bilinear pixels/clock, or four 2:1 anisotropic filtered pixels/clock, or four FP16 bilinear-filtered pixels/clock. Total bilinear texture addressing and filtering capability for an entire high-end GeForce GTX 200 GPU is 80 pixels per clock.

GeForce GTX 200 GPUs employ a more efficient scheduler, allowing the chips to attain close to theoretical peak performance in texture filtering. In real world measurements, it is 22% more efficient than the GeForce 9 Series.

Chip	Theoretical Bilinear Fillrate	Measured Rate (3DMark multitex)	Measured Performance / Theoretical Performance
GeForce 9 Series	33,600	25,600	76.2%
GeForce GTX 200 GPUs	51,840	48,266	93.1%

Table 4: Theoretical vs Measured Texture Filtering Rates

Higher Shader to Texture Ratio

Because games and other visual applications are continually employing more and more complex shaders, the GeForce GTX 200 GPU design shifts the balance to a higher shader to texture ratio. By adding one more SM to each TPC, and keeping texturing hardware constant, the shader to texture ratio is increased by 50%. This shift allows the GeForce GTX 200 GPUs to perform efficiently for both today's and tomorrow's games.

ROP Improvements

The previous-generation GeForce 8 series ROP subsystem supported multisampled, supersampled, transparency adaptive, and coverage sampling antialiasing. It also supported frame buffer (FB) blending of floating-point (FP16 and FP32) render target surfaces, and either type of FP surface could be used in conjunction with multisampled antialiasing for outstanding HDR rendering quality.

The new GeForce GTX 200 GPU ROP subsystem supports all of the previous generation features, and delivers a maximum of 32 pixels per clock output, equating to 4 pixels/clock per ROP partition × 8 partitions. Up to 32 color and Z samples per clock for 8 × MSAA are supported per ROP partition. Pixels using U8 (8-bit unsigned integer) data format can be blended at twice the rate per TPC of the oldergeneration GPUs. Given the prior generation GPU had six ROP partitions, it could output 24 pixels/clock and blend 12 pixels/clock. In contrast the GeForce GTX 280 can output and blend 32 pixels/clock.

1 GB Framebuffer

Today's 3D games use a variety of textures to attain realism. Normal maps are used to enhance surface realism, cubemaps for reflections, and high-resolution perspective shadow maps for soft shadows. This means much more memory is needed to render a single scene than classic rendering which relied mainly on the base texture. Deferred rendering engines also make extensive use of multiple render targets, where attributes of the image are rendered off screen before the final image is composed. These techniques consume an immense amount of video memory and memory bandwidth, especially when used in conjunction with antialiasing.

The GeForce GTX 280 and GeForce GTX 260 support 1,024 MB and 896 MB of frame buffer respectively, a two-fold improvement from over prior generation GPUs. With 1 GB of frame buffer, high-resolution antialiasing performance is dramatically improved. For example, deferred rendered games like S.T.A.L.K.E.R. can now be enjoyed with antialiasing.

Geometry Shading and Stream Out

Internal output buffer structures have been significantly upsized by a factor of 6× in GeForce GTX 200 GPUs compared to the prior generation, providing much faster geometry shading and stream out performance. Figure 8 shows the latest RightMark 3D 2.0 benchmark results, including geometry shading tests. The GeForce GTX 280 GPU is significantly faster than prior generation NVIDIA GPUs and competitive products.

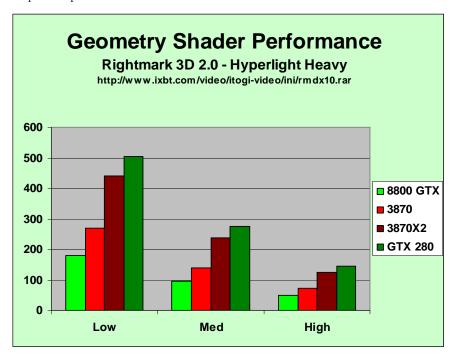


Figure 8: Geometry Shading Performance

Our own Medusa demo is highly dependent on the much faster geometry and stream out performance.

512-bit Memory Interface

Maximum memory interface width is expanded from 384 bits in previous-generation GPUs to 512 bits in GeForce GTX 200 GPUs, using eight 64-bit-wide frame buffer interface units. Memory bandwidth has been significantly increased.

In terms of rebalancing the architecture versus prior generations, the texture to frame buffer (TEX:FB) bandwidth ratio has also been modified to best support current and future workloads. NVIDIA engineers tested many applications to arrive

at the right balance of frame buffer bandwidth required to keep the texture units fully utilized and not starved for data.

General frame buffer efficiency has been improved for GeForce GTX 200 GPUs. We reworked the critical paths in the frame buffer to allow higher speed memory operation, up to 1.1 GHz GDDR3 stock speed. Memory bank access patterns and caching algorithms have also been improved. Additional compression hardware in GeForce GTX 200 GPUs effectively increase frame buffer bandwidth by permitting more data to traverse the interface per unit time, enabling better performance at higher resolutions.

Power Management Enhancements

GeForce GTX 200 GPUs include a more dynamic and flexible power management architecture than past generation NVIDIA GPUs. Four different performance / power modes are employed:

- ☐ Idle/2D power mode (approx 25 W)
- □ Blu-ray DVD playback mode (approx 35 W)
- □ Full 3D performance mode (varies—worst case TDP 236 W)
- HybridPowerTM mode (effectively 0 W)

Using a HybridPower-capable nForce motherboard, such as those based on the nForce 780a chipset, a GeForce GTX 200 GPU can be fully powered off when not performing intensive graphics operations and graphics output can be handled by the motherboard GPU (mGPU).

For 3D graphics-intensive applications, the NVIDIA driver can seamlessly switch between the power modes based on utilization of the GPU. Each of the new GeForce GTX 200 GPUs integrates utilization monitors ("digital watchdogs") that constantly check the amount of traffic occurring inside of the GPU. Based on the level of utilization reported by these monitors, the GPU driver can dynamically set the appropriate performance mode (i.e., a defined clock and voltage level) that minimizes the power draw of the graphics card—all fully transparent to the end

The GPU also has clock-gating circuitry, which effectively "shuts down" blocks of the GPU which are not being used at a particular time (where time is measured in milliseconds), further reducing power during periods of non-peak GPU utilization.

All this enables GeForce GTX 200 graphics cards to deliver idle power that is nearly 1/10th of its maximum power (approximately 25 W on GeForce GTX 280 GPUs). This dynamic power range gives you incredible power efficiency across a full range of applications (gaming, video playback, surfing the web, etc).

Many other areas of the GeForce GTX 200 GPU pipeline have been reworked to improve performance and reduce various processing bottlenecks.

Additional Pipeline and Architecture Enhancements

Starting from the top of the GeForce GTX 200 GPUs, the front-end unit communicates with the graphics driver running on the host system to accept commands and data. The communication protocol and certain software classes have

been modified to improve efficiency of data transfer between the driver and the front end.

The memory crossbar between the data assembler and the frame buffer units has been optimized, allowing the GeForce GTX 200 GPUs to run at full speed when performing indexed primitive fetches (unlike the prior generation which suffered some contention between the front end and data assembler).

The post-transform cache size has been increased, resulting in fewer pipeline stalls and faster communication from the geometry and vertex stages to the viewport clip/cull stage. (Setup rates are similar to prior generation, supporting up to one primitive per clock).

Z-Culling performance has also been improved, especially at high resolutions. Early-Z rejection rates have been increased because the number of ZROPs was increased. The maximum ZROP cull rate is 256 samples/clock or 32 pixels/clock.

GeForce GTX 200 GPUs also include significant micro-architectural improvements in register allocation, instruction scheduling, and instruction issue. The GPUs can now feed the execution units more swiftly. These improvements are responsible for the ability to dual-issue instructions to SPs and SFUs as previously discussed. Scheduling of work between texture units and the SM controller has also been improved.

Summary

NVIDIA's second generation unified visual computing architecture as embodied in the new GeForce GTX 200 GPUs is a significant evolution over the original unified architecture of GeForce 8 and 9 series GPUs. Numerous extensions and functional enhancements to the architecture permit a performance increase averaging 1.5× the prior architecture. Improvements in sheer processing power combined with improved architectural efficiency allow amazing speedups in gaming, visual computing, and high-end computation.

Compared to earlier GPUs such as GeForce 8800 GTX, the GeForce GTX 280 provides:

- ☐ 1.88× more processing cores
- □ 2.5× more threads per chip
- Doubled register file size
- ☐ Double-precision floating-point support
- Much faster geometry shading
- □ 1 GB frame buffer with 512-bit memory interface
- ☐ More efficient instruction scheduling and instruction issue
- ☐ Higher clocked and more efficient frame buffer memory access
- ☐ Improvements in on-chip communications between various units
- Improved Z-cull and compression supporting higher performance at high resolutions, and
- □ 10-bit color support

These all result in enough graphics and compute power to deliver the most intensive and extreme 3D gaming experiences and teraflop performance for demanding highend compute-intensive applications.

NVIDIA SLI technology is taken to new levels with GeForce GTX 200 GPUs and NVIDIA PhysX technology will add amazing new graphical effects to upcoming game titles. CUDA applications will benefit from additional cores, far more threads, double-precision math, and increased register file size.

Wise users purchasing new systems will conduct performance analyses to optimize their PC architecture. They will find that a lower-end CPU paired with a higher-end GPU produces more performance than the reverse and for the same price. This heterogeneous computing using the right processors for the right tasks and designing optimized PCs to take advantage of it is the wave of the future.

Appendix A: Retrospective

Over the past decade, NVIDIA's graphics processing units (GPUs) have evolved from specialized, fixed-function 3D graphics processors to highly programmable, massively multithreaded, parallel-processing architectures used for visual computing and high-performance computation.

NVIDIA GeForce GPUs enable incredibly realistic 3D gaming and outstanding high-definition video playback, while NVIDIA Quadro[®] GPUs provide the highest quality and fastest workstation graphics for professional design and creation. For high-performance computing tasks in various engineering, scientific, medical, and financial fields, NVIDIA's new Tesla[™] GPUs and CUDA parallel programming environment enable supercomputing-level performance on the desktop, at a fraction of the cost of comparably performing CPU-based multiprocessor clusters.

The GeForce 8800 GPU was launched in November 2006. It was the world's first DirectX 10 GPU with a unified shader architecture. This was important as each of the unified shader processing cores could be dynamically allocated to vertex, pixel, and geometry workloads, making it far more efficient than prior-generation GPUs, which used a fixed number of pixel processing units and a fixed number of vertex processing units. This same unified architecture provided the framework for efficient high-end computation using NVIDIA CUDA software technology.

The GeForce 9 Series GPUs were introduced in 2007, offering a vastly improved price-performance ratio and advanced PureVideo® features. Its smaller chip allowed dual-GPU GeForce 9800 GX2 graphics boards to be built more efficiently, while offering up to twice the performance of the GeForce 8800 GTX.

As of May 2008, over 70 million NVIDIA GeForce 8 and 9 Series GPUs have shipped and each supports CUDA technology, allowing greatly accelerated performance for mainstream visual computing applications like audio and video encoding and transcoding, image processing, and photo editing. These GPUs also support the new NVIDIA PhysX technology for enabling real-time physics in games.

GPUs are the most important and most powerful processors in the new era of visual computing. High-end GeForce GTX 200 GPUs provide the best user experience when running intensive DirectX 10-based games like *Crysis* at high quality and high resolution settings. Very capable motherboard and mid-range GPUs are also needed for stutter-free, high-definition video playback on the PC while simultaneously displaying the Aero 3D user interface of Windows Vista.

Appendix B: Figure 3 References

- 1. "Interactive Visualization of Volumetric White Matter Connectivity in DT-MRI Using a Parallel-Hardware Hamilton-Jacobi Solver," by Won-Ki Jeong, P. Thomas Fletcher, Ran Tao, and Ross T. Whitaker
- "GPU Acceleration of Molecular Modeling Applications."
- 3. Video encoding uses iTunes on the CPU and Elemental on the GPU running under Windows XP. CPUs tested were Intel Core 2 Duo 1.66 GHz and Intel Core 2 Quad Extreme 3 GHz. GPUs tested were GeForce 8800M on the Gateway P-Series FX notebook and GeForce 8800 GTS 512 MB. CPUs and GeForce 8800 GTS 512 were run on Asus P5K-V motherboard (Intel G33 based) with 2 GB DDR2 system memory. Based on an extrapolation of 1 min 50 sec 1280 × 720 high-definition movie clip.
- 4. http://developer.nvidia.com/object/matlab_cuda.html
- 5. "High performance direct gravitational N-body simulations on graphics processing units paper," communicated by E.P.J. van den Heuvel
- 6. "LIBOR," by Mike Giles and Su Xiaoke.
- 7. "FLAG@lab: An M-script API for Linear Algebra Operations on Graphics Processors."
- 8. http://www.techniscanmedicalsystems.com/
- 9. "General Purpose Molecular Dynamics Simulations Fully Implemented on Graphics Processing Units," by Joshua A. Anderson, Chris D. Lorenz, and A. Travesset
- 10. "Fast Exact String Matching On the GPU," presentation by Michael C. Schatz and Cole Trapnell

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, Quadro, Tesla, CUDA, PhysX, nForce, PureVideo, and SLI are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated

Copyright

© 2008 NVIDIA Corporation. All rights reserved.



Intel® Processor Graphics DirectX* Developer's Guide

How to maximize graphics performance on Intel® microarchitecture codename Sandy Bridge

Copyright © 2008-2010 Intel Corporation

All Rights Reserved

Document Number: 321371-002

Revision: 2.9.6

Contributors: Jeff Freeman, Chris McVay, Chuck DeSylva, Luis Gimenez, Katen Shah, Jeff Frizzell, Ben Sluis, Anthony Bernecky, Raghu Muthyalampalli, Ganeshkumar

Doraisamy, Steven Smith, Axel Mamode

World Wide Web: http://www.intel.com

Document Number: 321671-005US

ATI 2091 LG v. ATI IPR2015-00326



Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Software Source Code Disclaimer

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license:

Intel Sample Source Code License Agreement

This license governs use of the accompanying software. By installing or copying all or any part of the software components in this package, you ("you" or "Licensee") agree to the terms of this agreement. Do not install or copy the software until you have carefully read and agreed to the following terms and conditions. If you do not agree



to the terms of this agreement, promptly return the software to Intel Corporation ("Intel").

Definitions:

- A. "Materials" are defined as the software (including the Redistributables and Sample Source as defined herein), documentation, and other materials, including any updates and upgrade thereto, that are provided to you under this Agreement.
- B. "Redistributables" are the files listed in the "redist.txt" file that is included in the Materials or are otherwise clearly identified as redistributable files by Intel.
- C. "Sample Source" is the source code file(s) that: (i) demonstrate(s) certain functions for particular purposes; (ii) are identified as sample source code; and (iii) are provided hereunder in source code form.
- D. "Intel's Licensed Patent Claims" means those claims of Intel's patents that (a) are infringed by the Sample Source or Redistributables, alone and not in combination, in their unmodified form, as furnished by Intel to Licensee and (b) Intel has the right to license.
- 2. License Grant: Subject to all of the terms and conditions of this Agreement:
 - A. Intel grants to you a non-exclusive, non-assignable, copyright license to use the Material for your internal development purposes only.
 - B. Intel grants to you a non-exclusive, non-assignable copyright license to reproduce the Sample Source, prepare derivative works of the Sample Source and distribute the Sample Source or any derivative works thereof that you create, as part of the product or application you develop using the Materials.
 - C. Intel grants to you a non-exclusive, non-assignable copyright license to distribute the Redistributables, or any portions thereof, as part of the product or application you develop using the Materials.
 - D. Intel grants Licensee a non-transferable, non-exclusive, worldwide, non-sublicenseable license under Intel's Licensed Patent Claims to make, use, sell, and import the Sample Source and the Redistributables.

3. Conditions and Limitations:

- This license does not grant you any rights to use Intel's name, logo or trademarks.
- B. Title to the Materials and all copies thereof remain with Intel. The Materials are copyrighted and are protected by United States copyright laws. You will not remove any copyright notice from the Materials. You agree to prevent any unauthorized copying of the Materials. Except as expressly provided

Intel® Processor Graphics DirectX* Developer's Guide



herein, Intel does not grant any express or implied right to you under Intel patents, copyrights, trademarks, or trade secret information.

- C. You may NOT: (i) use or copy the Materials except as provided in this Agreement; (ii) rent or lease the Materials to any third party; (iii) assign this Agreement or transfer the Materials without the express written consent of Intel; (iv) modify, adapt, or translate the Materials in whole or in part except as provided in this Agreement; (v) reverse engineer, decompile, or disassemble the Materials not provided to you in source code form; or (vii) distribute, sublicense or transfer the source code form of any components of the Materials and derivatives thereof to any third party except as provided in this Agreement.
- 4. No Warranty:

THE MATERIALS ARE PROVIDED "AS IS". INTEL DISCLAIMS ALL EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO THEM, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR ANY PARTICULAR PURPOSE.

- 5. Limitation of Liability: NEITHER INTEL NOR ITS SUPPLIERS SHALL BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.
- 6. USER SUBMISSIONS: You agree that any material, information or other communication, including all data, images, sounds, text, and other things embodied therein, you transmit or post to an Intel website or provide to Intel under this Agreement will be considered non-confidential ("Communications"). Intel will have no confidentiality obligations with respect to the Communications. You agree that Intel and its designees will be free to copy, modify, create derivative works, publicly display, disclose, distribute, license and sublicense through multiple tiers of distribution and licensees, incorporate and otherwise use the Communications, including derivative works thereto, for any and all commercial or non-commercial purposes
- 7. TERMINATION OF THIS LICENSE: This Agreement becomes effective on the date you accept this Agreement and will continue until terminated as provided for in this Agreement. Intel may terminate this license at any time if you are in breach of any of its terms and conditions. Upon termination, you will immediately return to Intel or destroy the Materials and all copies thereof.
- 8. U.S. GOVERNMENT RESTRICTED RIGHTS: The Materials are provided with "RESTRICTED RIGHTS". Use, duplication or disclosure by the Government is subject to restrictions set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor. Use of the Materials by the Government constitutes acknowledgment of Intel's rights in them.



9. APPLICABLE LAWS: Any claim arising under or relating to this Agreement shall be governed by the internal substantive laws of the State of Delaware, without regard to principles of conflict of laws. You may not export the Materials in violation of applicable export laws.

Revision History

Document Number	Revision Number	Description	Revision Date
321671-001US	1.0	Re-drafted for Intel® 4-Series Chipsets.	Sept 2008
321671-001US	1.1	Re-drafted for Intel® 4-Series Chipsets.	Sept 2008
321671-002US	2.6.6	Intel HD Graphics DirectX* Developer's Guide (Sandy Bridge).	March 2009
321671-003US	2.6.7	Intel® HD Graphics DirectX* Developer's Guide (Sandy Bridge).	April 2009
321671-004US	2.7.1	Intel® HD Graphics DirectX* Developer's Guide (Sandy Bridge) featuring Intel® HD Graphics	Feb 2010
321671-005US	2.8.0	Intel® HD Graphics DirectX* Developer's Guide for the Intel® microarchitecture codename Sandy Bridge.	August 2010
321671-006US	2.9.5	Incremental changes and additions related to Intel microarchitecture codename Sandy Bridge	December 2010
	2.9.6	Incremental changes prior to GDC release	February 2011

^{*} Other names and brands may be claimed as the property of others. Copyright (C) 2008 – 2010, Intel Corporation. All rights reserved.



Contents

6

Disclaimer an	d Legal	Information	2
Software Sou	rce Cod	e Disclaimer	2
Revision Histo	ory		5
1	About 1	this Document	8
	1.1	Intended Audience	8
	1.2	Conventions, Symbols, and Terms	
Table 1 Codin	ng Style	and Symbols Used in this Document	8
Table 2 Terms	s Used i	n this Document	9
	1.3	Further Help Beyond this Guide	9
2	Intel®	HD Graphics to Intel® Processor Graphics	10
	2.1	Intel® HD Graphics is Becoming Core	
	2.2	What's New in Intel® Processor Graphics	
	2.3	Intel® Processor Graphics Specifications	
3	Quick 7	Tips: Graphics Performance Tuning	15
		3.1.1 Optimizing for the Vertex Cache	15
		3.1.2 Other Tips On Vertex/Primitive Processing	
	3.2	Shader Capabilities	
		3.2.1 DirectX* 11 on 10	18
		3.2.2 Tips on Shader Capabilities	18
	3.3	Texture Sample and Pixel Operations	20
		3.3.1 Tips on Texture Sampling / Pixel Operations	21
	3.4	Microsoft DirectX* 10 Optional Features	22
	3.5	Managing Constants on Microsoft DirectX*	22
		3.5.1 Tips on Managing Constants on Microsoft DirectX* 9	
		3.5.2 Tips on Managing Constants on Microsoft DirectX* 10	
	3.6	Advanced DirectX* 9 Capabilities	24
		3.6.1 FourCC and other surface and texture formats	
		3.6.2 Notes on supported FourCC texture formats	
		3.6.3 MSAA Under DirectX* 9	
	3.7	Graphics Memory Allocation	
		3.7.1 Checking for Available Memory	
		3.7.2 Tips On Resource Management	
	3.8	Identifying Intel® Processor Graphics and Specifying Graphics Presets	
	3.9	Surviving a Graphics Hardware Switch on the Fly	
		3.9.1 Microsoft DirectX* 9 Algorithm	
		3.9.2 Algorithm for DirectX* 10	
	3.10	Some suggestions, tips & tricks from the field	
		3.10.1 Device Capabilities	
		3.10.2 DirectX* 9 Extensions	
		3.10.3 Revisit Assumptions on Performance	
		3.10.4 Avoid Writing to Uniocked Bullers	s ı

About this Document



	3.10.5 Avoid Tight Polling on Queries	31
4	Appendix: Sample Code for Identifying Intel® Processor Graphics and Specifying Graphics Presets	
5	Support	37
6	References	38



1 About this Document

This document provides development hints and tips to ensure that your customers will have a great experience playing your games and running other interactive 3D graphics applications on Intel® Processor Graphics. This document details software development practices using the latest generation of Intel processor graphics: Intel® Processor Graphics as well as two previous generations of the Intel® Graphics Media Accelerator with a focus on performance analysis using Microsoft DirectX*. Intel tools useful in optimizing graphics applications are introduced in a section detailing performance analysis with the Intel® Graphics Performance Analyzers (Intel® GPA).

Intel® Graphics is split into product generations. The latest one was introduced in 2011 with Intel® microarchitecture codename Sandy Bridge. This family of processors is now on the same silicon as the CPU and it is now called *Intel® Processor Graphics*. In addition to vastly improved performance, Intel Processor Graphics also adds significant new features and functionalities over the previous generation of Intel Graphics called Intel® HD Graphics. Intel Processor Graphics currently represents the most common graphics solution chosen by new PC purchasers. This document is meant to help you include this broad market as a target for your applications and optimize the experience for widest people. By following the tips and tricks in this document, you have the opportunity to make your application shine with the graphics volume market leader.

1.1 Intended Audience

This document is targeted at experienced graphics developers who are familiar with OpenGL*/Microsoft DirectX*, C/C++, multithreading and shader programming, Microsoft Windows* operating systems, and 3D graphics.

1.2 Conventions, Symbols, and Terms

Table 1 Coding Style used in the Document

```
Source code:
for( int i = 0; i < 10; ++i )
{
   cout << i << endl;
}</pre>
```

The following terms are used in this document.



Table 2 Terms Used in this Document

- HDG -Intel® HD Graphics, the generation of graphics from Intel which was characterized by the graphics subsystem being on a separate die from the CPU
- 2. **Processor Graphics** The latest generation of graphics from Intel® included in the same processor die of the Intel® microarchitecture codename Sandy Bridge family of processors
- 3. **UMA** Unified Memory Architecture an architecture where the graphics subsystem does not have exclusive dedicated memory and uses the host system's memory (SDRAM)
- 4. **DVMT** Dynamic Video Memory Technology a memory allocation scheme in UMA systems which allocates an exclusive, dynamically resizable chunk of main memory to the graphics (driver)
- 5. VF Vertex Fetch
- 6. VS Vertex Shader
- 7. PS Pixel Shader
- 8. **GS** Geometry Shader
- 9. **EU** Execution Unit, a vector machine component
- 10. Is Instruction cache
- 11. SO Stream Output
- 12. **HWVP** Hardware vertex processing see the following Intel document for more information on this: http://www.intel.com/assets/pdf/whitepaper/318888.pdf

1.3 Further Help Beyond this Guide

There are several other places you can look for additional information on Intel Processor Graphics, including the following sites:

Intel® HD Graphics (previous generation): http://software.intel.com/en-us/articles/intel-graphics-developers-quides/

Intel® 4 Series Chipsets (the Intel® 4500, X4500, and X4500HD GMAs) Developer's Guide: http://software.intel.com/en-us/articles/intel-graphics-media-accelerator-developers-quide/

Intel® 3 Series Express Chipsets including the Intel® 3000 GMA and Intel® X3000 GMA Developer's Guide: http://software.intel.com/en-us/articles/intel-gma-3000-and-x3000-developers-quide/.

We hope your questions are covered in these resources, including this Guide. We are constantly updating these resources and welcome your comments and suggestions. If you have questions not answered in these resources, or have suggestions on improving the Guide, please get in touch with us at: GameDevInput@intel.com. If you are actively working with Intel already, you can also reach us through your engineering or account management contacts.



2 Intel® HD Graphics to Intel® Processor Graphics

2.1 Intel® HD Graphics is Becoming Core

Several versions of the Intel® $Core^{TM}$ i3, $Core^{TM}$ i5, and $Core^{TM}$ i7 processors have launched in 2011 using Intel® microarchitecture codename Sandy Bridge. These represent the first instantiation of complete platform integration, with Intel® Processor Graphics co-residing on the CPU die.

Two key versions of graphics will be available, Intel® Processor Graphics 2000 and Intel® Processor Graphics 3000, with Intel® Processor Graphics 2000 targeting lower voltage (lower power) applications and Intel® Processor Graphics 3000 a more mainstream set of applications.



Table 3 Evolution of Intel® Processor Graphics

2009	2010	2011	
GMA Series 4 Chipset	Intel codename Ironlake - 1st Gen CPU Integration	Intel® microarchitecture codename Sandy Bridge 1st Gen CPU/Graphics single die	
DirectX* 10 SM4, OpenGL* 2.0	DirectX* 10, SM4, OpenGL* 2.1	DirectX* 10.1 SM4.1 , OpenGL* 3.0, DirectX* 11 API on DirectX* 10 hardware	
Mobile / Desktop*: 21 / 32 GFLOPs	Mobile / Desktop: 37 / 43 GFLOPs	Mobile / Desktop: ~105-125 GFLOPs for Processor Graphics 3000 & ~55-65 GFLOPS for Processor Graphics 2000	
Mobile/Desktop 400 - 533 MHz/ 800 MHz	Mobile/Desktop 500-766 MHz/ 533 - 900 MHz	Mobile/Desktop Base: 350-650 MHz/650-850 MHz Turbo: 900-1250 MHz/1100-1350 MHz	
	1.7x 3D performance increase	Intel® Processor Graphics 2000: 6 Execution Units ≥1x with lower voltage requirements.	Intel® Processor Graphics 3000: 12 Execution Units 1.5-2.5x performance increase
* Single precision peak values with MAD instructions.			



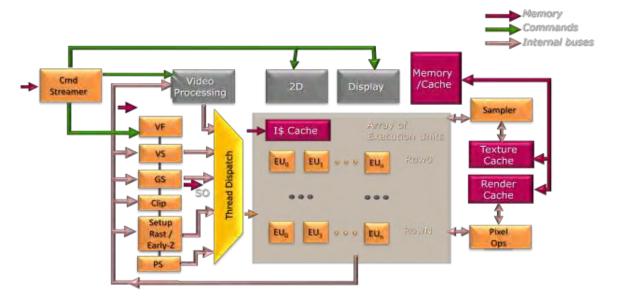


Figure 1 Intel® Processor Graphics Architecture Diagram

Intel® Processor Graphics has been architected to support Microsoft DirectX* 10.1 and take advantage of a generalized unified shader model including support for Shader Model 4.1 and lower. The platform also has support for DirectX* 11 on DirectX* 10 hardware. The graphics core executes vertex, geometry, and pixel shaders on the programmable array of Execution Units (EUs). The EUs have programmable SIMD (Single Instruction, Multiple Data) widths of 4 and 8 element vectors (two 4 element vectors paired) for geometry processing and 8 and 16 single data element vectors for pixel processing). Each EU is capable of executing multiple threads to cover latency. The new generation of Intel® Processor Graphics now integrates transcendental shader instructions into the EU units, rather than a shared math box found in prior generations, resulting in improved processing of instructions such as POW, COS, SIN. Clipping and setup have moved to Fixed Function units, further increasing performance by reducing contention within the EU's. The end result is the fastest Intel® Processor Graphics to date.

2.2 What's New in Intel® Processor Graphics

2.2.1.1 Graphics Features

The latest version of Intel® Processor Graphics includes several performance changes since the previous generation:



Table 4 Intel® Processor Graphics Feature Specifications

3D Pipeline	Key Improvements in Intel® Processor Graphics
Geometry Processing	 Improved throughput up to 2x better than previous generations Sharing of the last level cache with the CPU Increased number of threads for vertex shading Faster clip, cull, and setup Improved throughput of geometry shader and stream out OpenGL* driver now uses hardware geometry processing
Rasterization and Z	 Improved hierarchical Z performance Improved clear performance Added OpenGL* MSAA 4X support Added 2X and 4X MSAA support under DirectX* 9 and DirectX* 10
Computes	> 3X increase in transcendental computations Overall arithmetic performance improvement in shaders due to math box integration within EU's
Texture and Pixel Processing	 Added support for Gather4, Target Independent Blend Modes, Per Sample Shader Frequency, TextureCubeArray, VS/GS 32 Registers, per the DirectX* 10.1 specification Improved fill rate for short shaders due to fixed function setup management of barycentric coefficients

2.2.1.2 Intel® Turbo Boost Technology

Intel® Processor Graphics utilizes a dynamic frequency on mobile and desktop graphics to automatically increase the clock frequencies of the CPU cores and the graphics cores to boost performance when the workload demands it and also to scale back the frequencies when demand decreases. Intel® microarchitecture codename Sandy Bridge supports higher performance boosts after extended CPU idle periods. In addition to the Intel® Turbo Boost Technology on the CPU, a similar technology has been extended to graphics on both the mobile and desktop platforms. This allows the graphics subsystem to run at higher frequencies when the CPU is not using its nominal thermal design power (TDP). In combination, these technologies dynamically manage the CPU and graphics core performances based on workload demands, to allow for better performance when needed and minimize power usage when possible.



2.3 Intel® Processor Graphics Specifications

Table 5 Intel® Processor Graphics Feature Specifications

Intel Graphics Core	Intel® Processor Graphics				
Intel Chipset (see Error! Reference source not found.)	Intel® microarchitecture codename Sandy Bridge				
Graphics Architecture	Intel® Processor Graphics 3000 Intel® Processor Graphics 3000				
Segment	Desktop/Mobile Desktop/Mobile)/Mobile		
Maximum Video Memory	Depends on system memory and operating system. Windows Vista*/Windows* 7: refer to the table below: System Memory 1GB 2GB >2GB				
	Max Available Video Memory	256 IVIB		783 MB	1692 MB
DirectX* Support	10.1, DirectX* 11 on DirectX* 10 hardware, Compute Shader 4.x, DirectX* 11 API multi-threaded rendering on DirectX* 10 hardware (asynchronous object creation supported, software support for asynchronous command list in the runtime)				
OpenGL* Support	3.0				
Shader Model Support	4.1				



Quick Tips: Graphics Performance Tuning

3.1.1 Optimizing for the vertex cache

Use IDirect3DDevice9::DrawIndexedPrimitive (DirectX* 9) or ID3D10Device::DrawIndexed (DirectX* 10) to maximize reuse of the vertex cache.

Pre- and post-transform vertex cache sizes can vary significantly, even across different generations of Intel Graphics platforms. To maximize performance of vertex processing, it can help to optimize the ordering of vertices and triangle indices in your vertex and index buffers.

There are two methods you can use to optimize your data for the vertex cache:

The preferred method is to use the D3DXOptimizeFaces and D3DXOptimizeVertices APIs. These apply a good generalized optimization that will work well for all cache sizes (and across all hardware). Since this optimization works well across all hardware, it can be applied at authoring time (e.g. when exporting mesh data from your content creation pipeline), cutting down level load times.

The code snippet below demonstrates how to use these APIs on mesh data:

```
void OptimizeMesh( WORD * Indices, // [In/Out] - Index buffer data
                   DWORD NumFaces, // Number of faces in the mesh
                   Vertex * Vertices, // [In/Out] - Vertex buffer data
                   DWORD NumVertices ) // Number of vertices in the mesh
   HRESULT hr;
    // Create a "re-map" buffer for the new face ordering, and
    // calculate the new order.
   DWORD *Remap = new DWORD[ NumFaces ];
   hr = D3DXOptimizeFaces( IndicesIn, NumFaces, NumVertices,
                            FALSE, Remap );
   // Make a copy of the old indices, which we'll pull from for the new
    // re-ordered list of indices.
    WORD *OldIndices = new WORD[ NumFaces * 3 ];
    memcpy( OldIndices, IndicesIn, sizeof(WORD) * NumFaces * 3 );
    WORD * NewFaces = Indices;
    // Apply the new mapping.
    for( DWORD i = 0; i < NumFaces; ++i )</pre>
        WORD *OldFaceBase = OldIndices + ( Remap[ i ] * 3 );
        NewFaces[0] = OldFaceBase[0];
```

How to maximize graphics performance on Intel® Integrated Graphics

15



```
NewFaces[1] = OldFaceBase[1];
    NewFaces[2] = OldFaceBase[2];
    NewFaces += 3;
delete[] Remap;
delete[] OldIndices;
// Create a "re-map" buffer for the new vertex ordering, and
// calculate the new order.
DWORD ActualVertexCount = 0;
Remap = new DWORD[ NumVertices ];
hr = D3DXOptimizeVertices( Indices, NumFaces, NumVertices,
              FALSE, Remap );
// Count how many vertices we actually have. Remap indices of
// Oxffffffff indicate a vertex that was not referenced by any faces.
DWORD dwVertexCount = 0;
for( DWORD i = 0; i < NumVertices; ++i )</pre>
    if( aRemap[ i ] == 0xffffffff )
        break;
    ++ActualVertexCount;
    // Copy the vertex data into a temp buffer.
Vertex *OldVertices = new Vertex[ NumVertices ];
memcpy( OldVertices, Vertices, sizeof(Vertex) * NumVertices );
// Perform the remapping
const DWORD *CurrentRemap = Remap;
const SVertex *OldVertex = OldVertices;
for( DWORD i = 0; i < ActualVertexCount; ++i )</pre>
    aVertices[ *( CurrentRemap++ ) ] = *( OldVertex++ );
// We'll also need to re-index our vertices to point to the new
// vertex locations.
for( DWORD i = 0; i < NumFaces * 3; i++ )</pre>
    Indices[ i ] = ( WORD )aRemap[ Indices[ i ]];
delete[] Remap;
delete[] OldVertices;
```

3.1.2 Other Tips On Vertex/Primitive Processing

- 1. Ensure adequate batching of primitives to amortize runtime and driver overhead.
 - a. Use instancing to enable better vertex throughput especially for small batch sizes. This also minimizes state changes and Draw calls.



- b. Aim for 2000 or fewer draw-calls per frame (or less than 50,000 draws per second). Above this number the CPU overhead in the driver can become prohibitive.
- Minimize render state changes between batches to reduce the number of pipeline flushes.
- 2. Use static vertex buffers where possible.
- 3. Use visibility tests to reject objects that fall outside the view frustum to reduce the impact of objects that are not visible.
 - a. Set D3DRS CLIPPING to FALSE for objects that do not need clipping.
- 4. Take advantage of Early-Z rejection.
 - a. Render with a Z-only pass (meaning no color buffer writes or pixel shader execution) followed by a normal render pass. This uses the higher performance of Early-Z to reject occluded fragments which reduces compute times and raster operations.
 - b. Balance a Z-only pass against the inherent cost of such an additional pass. A Z-only pass increases the number of draw calls (which can impact the CPU usage) as well as the amount of work done up to the Rasterizer. Measure the performance difference between the two approaches to assess the actual benefit.
 - c. Avoid writing Z values (depth) in the pixel shader. Writing the depth value will skip the Early-Z hardware optimization algorithm since it potentially changes the visibility of the fragment.
- 5. Use the Occlusion Query feature of Microsoft DirectX* to reduce overdraws for complex scenes. Render the bounding box or a simplified model if it returns zero, then the object does not need to be rendered at all.
 - a. Allow sufficient time between Occlusion Query tests and verifying the results to avoid serializing the platform. See the Microsoft DirectX* 10 "Draw Predicated" sample included in the Microsoft DirectX* SDK for more information on this.
 - See Section 3.10.5 Avoid Tight Polling on Queries for more tips on using queries properly.
- 6. Consider drawing the opaque overlays in the scene such as heads up displays (HUD) first and writing them to the Z buffer. This reduces the screen rendering area leading to considerable performance improvement.

3.2 Shader Capabilities

Intel® Processor Graphics includes support for Microsoft DirectX* Shader Model 4.1 for 10.1 devices and Shader Model 3.0 for DirectX* 9 devices. Intel® microarchitecture codename Sandy Bridge provides significantly improved computational capability and better handling of large and complicated shaders over prior architectures.



3.2.1 DirectX* 11 on 10

In addition to DirectX* 10.1 hardware support, Intel® Processor Graphics includes partial DirectX11 API support. This allows developers to use the DirectX* 11 API on DirectX* 10 (or 10.1) level hardware. There are two DirectX11-specific features exposed through this paradigm, specifically:

- Multithreading: The D3D11 API provides means to record command lists on multiple threads (multithreaded rendering) as well as object creation on multiple threads. The Processor Graphics driver currently supports driver-level concurrent object creation.
- 2. Compute Shader 4 (CS4): A limited subset of DirectX11-level compute shader (CS5) a number of restrictions, specifically:
 - Maximum threads is limited to 768.
 - Z dimension of numthreads/dispatch limited to 1.
 - Only one unordered access view bound per shader (and only RWStructuredBuffers and RWByteAddressBuffers are bindable).
 - Threads are limited to writing to only their own region of groupshared memory (although they can thread from any location).
 - SV_GroupIndex must be used when accessing groupshared memory for writing.
 - Groupshared memory limited to 16KB per group.
 - Atomics and double-precision are not available.

Other Direct X^* 11 features (Tessellation, CS5, extended texture formats such as BC6/7) are not supported.

 $\label{thm:condition} \begin{tabular}{ll} Utilize the DirectX* \ \ \ ID3D11Device:: CheckFormatSupport() \ \ API for individual format support of the D3D11_FORMAT_SUPPORT enumeration. \end{tabular}$

3.2.2 Tips on Shader Capabilities

- Reduce texture sampling intensive operations when possible. The following common shader effects typically affect performance and should be tested for performance and optimization. Pay special attention to full screen post processing effects including per-pixel and multiple pass techniques when evaluating graphics related performance bottlenecks.
 - a. Glow/Bloom
 - b. Motion blur
 - c. Depth of field
 - d. HDR/tone mapping
 - e. Heat distortion
 - f. Atmospheric effects
 - g. Dynamic Ambient Occlusion
 - h. High quality soft shadows
 - i. Parallax occlusion mapping with a wide radius
- 2. Balance texture samples and shader complexity.



- a. Recommend greater than 4:1 ratio of ALU: Sample for better latency coverage. A larger ratio may be better for floating point textures, higher order filtering, and 3D textures.
- 3. Space your texture sampling calls away from where it is used in pixel shaders when possible and practical to maximize EU utilization.
- 4. Generically speaking where a choice is available between using programmable shading and fixed function pipeline, programmable shading gives better performance.
 - In particular, use shader-based fog instead of fixed function fog on DirectX*
 Fixed Function fog has been deprecated on SM 3.0.
 - b. One notable exception to this rule is the use of the *texkill* instruction. Consider using alpha test to achieve the same result if possible, as it performs better on this platform.
- 5. Use flow control wisely it can be expensive. In some cases, it might be advantageous to split a single shader into multiple ones, to avoid flow control statements.
 - a. The pixel shader operates on up to 16 pixels in parallel. This means the benefits of dynamic flow control will depend on the likelihood of the number of pixels taking the same branch. If any 2 of those pixels take different paths, all pixels will execute both branches of the control flow.
 - b. Dynamic flow control can provide significant benefits by skipping a large number of computations. Ensure that this is used where a large number of instructions can be skipped.
- 6. Shader Model 3.0 supports dynamic flow instructions such as if bool, break, and break_comp. It also supports predication registers. Often the dynamic flow control instructions can be eliminated by using predication registers in their place. When possible, use predication instructions over dynamic flow control for shorter branching instruction sequences. Optimize your shader performance by adequate use of your processor graphics mask alpha if you are not using it.
- 7. Minimize the usage of geometry shaders, especially in cases where they result in geometry amplification (more geometry output from the GS than came in). If in doubt, examine your application's performance with and without geometry shaders to determine if they are a significant performance issue.
- 8. In general, minimize use of StreamOut and DrawAuto() for optimal performance, since they can incur a significant bandwidth penalty with their usage of system memory.



3.3 Texture Sample and Pixel Operations

Table 6 Intel® Processor Graphics Texture Sampling and Pixel Specifications

CPU Product	See Error! Reference source not found.
Graphics Architecture	Desktop and Mobile (Processor Graphics 2000 & 3000)
Format Support	16/32-bit fixed point
	16/32-bit floating point operations
Max # of Samplers	Up to 16
Vertex Textures	Yes
Max 2D/3D/Cube Textures	8Kx8K/2Kx2Kx2K/8K
Filtering Type Support	BLF, TLF and Dynamic AF with up to 16 degrees of anisotropic filtering + DirectX* 10.1
Texture Compression	DirectX* 9: DXT1/3/5, ATI1, ATI2; DirectX* 10: BCx
Multi-Sample Render	MSAA 4X
Multi-Target Render	8
Alpha-Blend FP formats	FP16 and FP32 formats are supported. For a complete list, do a caps check on DirectX* 9 and on DirectX* 10, utilize the DirectX* CheckFormatSupport() call as format support may be added in future driver versions.



Table 7 Intel® Processor Graphics Sampler Filtering Specifications

Product	See Error! Reference source not found.			
32-bit Texels (per clock cycle) e.g. RGBA UNORM8, RG FP16, R FP32				
Point/Bilinear	1X			
Trilinear	1X			
Anisotropic (n samples)	0.5X/n			
64-bit Texels (per clock cycle) e.g. RG FP32, RGBA FP16, RGBA UINT16				
Point/Bilinear	Point/Bilinear 1X			
Trilinear 0.5X				
Anisotropic (n samples) 0.25X/n				
128-bit Texels (per clock cycle) e.g. RGBA FP32, RGBA UINT32				
Point	0.25X			
Bilinear	0.25X			
Trilinear 0.125X				

All sampler filtering types are supported, including dynamic anisotropic filtering.

3.3.1 Tips on Texture Sampling / Pixel Operations

- 1. Use compressed textures and mip-maps and minimize the use of large textures. Even though the architecture supports up to 8K×8K textures, for optimal performance it is best to use smaller textures.
- 2. Minimize the use of Anisotropic Filtering, and both Trilinear and Anisotropic filtering for floating point textures. With floating point texture formats, the performance of bilinear and trilinear filtering are not equivalent.



Examine the scene to determine where you can make performance/quality tradeoffs with texture filtering. Prefer bilinear filtering where there is little visual difference.

- Generically speaking, the more compact the texture format being used, the better the performance. DXT compressed formats are best. Minimize the use of 32-bit floating point textures, since they carry a heavy bandwidth penalty and fill the texture caches faster.
- 4. Use as few render targets as possible, ideally keeping it to less than four. More render targets requires more bandwidth. If in doubt, analyze your performance using a tool such as Intel® GPA to determine if you are fill bound.
- 5. Minimize the number of Clear calls. Clear surfaces, Color and Z/Stencil buffer at the same time when required.
- 6. Avoid stencil shadows as they are fill intensive.

3.4 Microsoft DirectX* 10 Optional Features

D3D10 specifies optional features that can be checked for support in the code through API functions like *CheckFormatSupport(...)*, *CheckMultipleQualityLevels(...)*, *CheckFeatureSupport(...)*

The current platform supports more of those optional features than the previous ones and even more features will likely be supported in future. So it is better to explicitly test for such features using those APIs rather than relying on vendor and device ID's for the platform.

For example, DirectX* 10 specifies a large number of resource types and data formats that are optional. Utilize the DirectX* 10 CheckFormatSupport(...) call to determine which ones are supported. Also, utilize the DirectX* 10 CheckFormatSupport(...) call for UNORM and SNORM blending support

The following optional features are supported at the time of review of this Guide:

- 1. MSAA 2X and 4X on DirectX* 10.
- 2. 32-bit floating point blending

3.5 Managing Constants on Microsoft DirectX*

Constants are external variables passed as parameters to the shaders; their values remain "constant" during each invocation of the shader program. Despite their name, constants are one of the most frequently changing values in a Microsoft DirectX*



application. A shader program can initialize a constant value statically to a value in the shader file or at runtime through the application.

Many of the recommendations described here are standard in the industry. They are very much applicable to Intel processor graphics and the recommendations attempt to detail them in a cohesive manner.

In addition to these points it is worth noting that care should be taken when porting from Microsoft DirectX* 9 to Microsoft DirectX* 10 to maintain performance. For more information on this topic, see the Intel publication "DirectX* Constants Optimizations For Intel® Processor graphics" [2] available on the Intel Software Network.

3.5.1 Tips on Managing Constants on Microsoft DirectX* 9

- The driver optimizes access to the most frequently used constants. Use less than 32 (FLOAT4) constants per shader to achieve the best performance gain from this feature. Limit the use of dynamic indexed constants (C[ax], C[r]) as these cannot be optimized by the driver and will result in high latency in shaders (dynamic indexed constants are normally found in vertex shaders).
- Prefer local constants over global constants the former are better for performance.
- 3. Immediate constants provide better performance than dynamic indexed constants. In dynamic indexed constants the driver cannot determine a previous index value and needs to create a full size constant buffer space in memory, instead of using the hardware constant buffer.

3.5.2 Tips on Managing Constants on Microsoft DirectX* 10

- As previously detailed for DirectX* 9 above, the driver optimizes access to the
 most frequently used constants. The same advice applies for DirectX* 10: use
 less than 32 constants per shader to achieve the best performance gain from this
 feature, and limit the use of dynamic indexed constants (C[ax], C[r]) as these
 cannot be optimized by the driver.
- 2. For better performance prefer multiple, smaller constant buffers. The constant buffers need to be loaded to the graphics subsystem ahead of the shader execution, and the entire buffer needs to reloaded every time its contents change. Larger the size of the buffer, longer it takes to load the buffer to the graphics subsystem, causing significant performance impacts. If multiple buffers are combined into a single larger buffer, every time any of the contents changes, the entire large buffer will have to be loaded again, degrading the performance. And, because of the performance impact that reloading a constant buffer can have, where possible, if multiple shaders share the same buffer that could help the performance. Whether that performance gain is actually realized depends on whether the shared buffer is still resident when the second shader sharing the buffer is executed. But sharing the buffers between shaders can help but does not hurt the performance. For optimal constant buffer management, smaller packed constant buffers grouped by frequency of update and access pattern are



ideal for higher performance. As an example: organize Per Frame/ Per Pass/ Per Instance constant buffers first which tend to be smaller in size and have a low update rate followed by Per Draw/Per Material constant buffers which may also be small but have a higher update rate. Put large constant buffers like skinning constants last.

- 3. If there are constants that are unused by most of the shaders move those to the bottom of the constant definition list so that you can bind a smaller buffer to those shaders
- 4. Break up constant buffers based on features that are optional in games (e.g. shadows, post-processing effects, etc.). Very likely, due to performance constraints for integrated platforms, some of these features are either going to be disabled or run with a lower setting. So it would be beneficial to break up the constants into separate buffers and then selectively disable the updates to these constant buffers based on the settings selected by the user.
- 5. When using indexed constant buffers, it is recommended to keep the buffer size tailored to actual needs. For example, if the shader iterates over five elements only, declare a 5-element constant buffer for this shader rather than a general purpose 50-element constant buffer shared among shaders. This allows the driver to optimize placement so that it incurs a low latency path.

3.6 Advanced DirectX* 9 Capabilities

Several advanced features beyond those required by the DirectX* 9 specifications are supported by the Intel Graphics Platforms. This section provides the details on some of them.

The feature list is current for Intel® microarchitecture codename Sandy Bridge, when this Guide is being prepared and is likely to be extended over time. Further, support for the features vary between different Intel platforms. We strongly advise the developers to first confirm in their code that the feature of interest is supported and provide alternative execution paths in the code where features are not supported.

These are capabilities not directly exposed by DirectX* 9 interfaces. Developers will have to use indirect methods to check for their availability. Code segments showing how to check for such features are given in this section.

3.6.1 FourCC and other surface and texture formats

Intel Graphics platforms support multiple surface-formats beyond those required by the DirectX* 9 specifications. These are listed in Table 8.



Table 8 Additional DirectX* 9 texture and surface formats supported on Intel platforms

Format	Resource Type	Usage	Description
INTZ	Texture	Depth/Stencil	For reading depth buffer as a texture
DF16	Texture	Depth/Stencil	For reading depth buffer as a texture
RESZ	Surface	Render Target	For converting a multisampled depth buffer into a depth stencil texture.
ATI1N	Texture	Texture	Single Channel Texture Compression. Functionally equivalent (but not identical) to DirectX* 10 BC4 format.
ATI2N	Texture	Texture	Two-channel Texture Compression. Functionally equivalent (but not identical) to DirectX* 10 BC5 format.
NULL	Texture	Render Target	Render Target with no memory allocated. Useful as a dummy render target to render exclusively to a depth buffer.
ATOC	Surface	Render Target	Alpha to coverage multisampling.

The following code segment shows a sample outline and works for most FourCC

formats.

See the code accompanying this Guide for an example of how to test for more formats.



3.6.2 Notes on supported FourCC texture formats

1. INTZ is a depth texture format meant for accessing 24-bit depth buffer and 8-bit stencil buffer. In addition to depth operations this allows for using it for stencil operations as well.

This surface cannot be used as a texture when it is being used for depth buffering, unless the depth writes are disabled.

- 2. DF24 and DF16 formats are meant for use in Percentage-Closer Filtering (PCF) used in shadow mapping applications that use PCF. DF16 has better performance.
- 3. Intel® microarchitecture codename Sandy Bridge supports MSAA under DirectX*
- 9. RESZ provides the ability to use multisampling in depth buffer and then copy the result as a single value into a depth-stencil buffer. This process is often referred to as "resolving" a multi-sample depth-stencil buffer into a single-sample depth-stencil buffer.
- 4. Some ISV's use FourCC formats at times known as ATI1N which allows for the compression of single channel textures. The latest Intel platforms support this for the benefit of those ISV's.
- 5. 3DC format is also known in the industry as ATI2N format. This format is also supported in the latest Intel platforms.
- 6. A *NULL Render Target* is a dummy render target format which acts like a valid render target with a crucial difference that the driver will not allocate any memory for it. DX9 requires that a color render target be used for *all* rendering operations. Since a color render target is often not used with depth-only render targets, using a NULL render target in such cases can avoid memory allocation for a color render target will not be used.
- 7. ATOC, short for "Alpha To Coverage", is meant for interpreting the alpha channel value for approximating the multichannel pixel coverage. The texture itself has no practical use; but you use it as an argument in the <code>SetRenderState(...)</code> function, to convert the incoming alpha value output by the pixel shader into multichannel pixel coverage value.

3.6.3 MSAA Under DirectX* 9

2x and 4X MSAA are supported in DX9 on the latest Intel platforms. In general MSAA requires the graphics engine to do more work and hence impacts the performance to varying degrees. The developers should weigh in the tradeoffs ahead of using MSAA in their titles.

3.7 Graphics Memory Allocation

Processor graphics will continue to use the Unified Memory Architecture (UMA) and Dynamic Video Memory Technology (DVMT) as noted in the chart below. As with past processor graphics solutions, UMA specifies that memory resources can be used for video memory when needed. DVMT is an enhancement of the UMA concept, wherein



system memory is allocated for balanced graphics and system performance. DVMT dynamically responds to system requirements and application demands, by allocating the proper amount of display, texturing, and buffer. The operating system views the Intel graphics driver as an application, which uses system memory to request allocation of additional memory for 3D applications, and returns the memory to the operating system when no longer required.

Table 9 Intel® Processor Graphics Memory Specifications

CPU Product	See Error! Reference source not found.	
Segment	Processor Graphics 2000	Processor Graphics 3000
Memory BW (GB/s)	17.1 - 21.3 17.1-25.6	
UMA Capability	2x DDR3 up to 1333	2x DDR3 up to 1600
Max DVMT (XP) 1 or 2GB System Memory	Limited to 1GB max for all system memory configurations	
Max DVMT (Windows Vista*/Windows* 7) x86/x64: System Memory	The memory is managed by the operating system and the driver.	
Memory Interface	64 bits	

3.7.1 Checking for Available Memory

Graphics applications often check for the amount of available free video memory early in execution. Developers typically want to know the amount of memory that the graphics device can access at full performance.

As a result of the dynamic allocation of graphics memory performed by the Intel® Processor Graphics devices (based on application requests), memory checks that only request the amount of 'local' or 'dedicated' graphics memory available do not supply a number that is appropriate for the Intel® Processor Graphics devices.

The Microsoft DirectX* SDK (June 2010) includes the VideoMemory sample code which demonstrates 5 commonly used methods to detect the total amount of video memory. Of these tests, GetVideoMemoryViaWMI is recommended. All other methods either return the local/dedicated graphics memory and consequently will report incorrect results on Intel® Processor Graphics, or will report the sum of the dedicated memory and the shared memory, something that is typically not suitable for discrete graphics devices. For more information, see the sample code: http://msdn.microsoft.com/en-us/library/ee419018(v=VS.85).aspx

3.7.2 Tips On Resource Management

 Allocate surfaces in priority order. The render surfaces that will be used most frequently should be allocated first. On Microsoft DirectX* 10, memory is taken care of for you by the OS. On Microsoft DirectX* 9:



- a. Use D3DPOOL DEFAULT for lockable memory (dynamic vertex/index buffers).
- b. Use D3DPOOL MANAGED for non-lockable memory (textures, back buffers, etc).
- 2. On D3D10, use of the <code>Copy...()</code> methods are preferred over calling the <code>Update...()</code> operations. Partial or sub-resource copies should be used sparingly. For example when updating all or most of the LODs of a resource use <code>CopyResource()</code> rather than multiple <code>CopySubResource()</code>.

3.8 Identifying Intel® Processor Graphics and Specifying Graphics Presets

Games often specify a range of graphics capabilities and presets to identify with a Low, Medium, and High fidelity level. Please refer to the Appendix for sample code that demonstrates how to identify Intel® Processor Graphics versions and set fidelity levels for older generations (Low) to the most recent (Medium) based on the requested D3D*_FEATURE_LEVEL.

Note that on Windows* 7, multiple graphics adapters are supported so care should be taken in determining which adapter will be used for rendering.

3.9 Surviving a Graphics Hardware Switch on the Fly

Intel in combination with third party graphics vendors jointly developed a switchable graphics solution that allows end users to switch on-the-fly, between two heterogeneous graphics hardware systems without a reboot. This functionality can incorporate the energy efficiency of Intel processor graphics with the 3D performance of discrete graphics hardware in a single notebook solution. This technology is applicable to about 30 million discrete graphics hardware notebooks purchased annually. Currently most applications running on PC platforms with heterogeneous graphics hardware do not survive when switched between the two at run-time as they do not re-query underlying graphics capability when the active adapter changes.

Keys to handling graphics hardware switches:

- New applications should be aware of multiple graphics hardware configurations and handle the D3DERR DEVICELOST and WM DISPLAYCHANGE messages properly.
- Legacy applications where possible should develop and distribute patches for old games to handle the D3DERR_DEVICELOST and WM_DISPLAYCHANGE messages.

3.9.1 Microsoft DirectX* 9 Algorithm

Microsoft DirectX* 9 applications should follow the below procedure to query GFX adapter's capabilities (re-create DX object/device) on D3DERR DEVICELOST:



- Manually save the current context including state and draw information in the application.
- 2. Query if the graphics adapter has changed, using the Windows* API's EnumDisplaySettings() or EnumDisplayDevices().
- 3. If the adapter has changed, then:
 - a. Recreate a Microsoft DirectX* device.
 - b. Restore the context.
 - c. Continue rendering from last scene rendered before the D3DERR_DEVICELOST event occurred.

3.9.2 Algorithm for DirectX* 10

By design, DirectX* 10 does not have the concept of device lost until the next Present, and the developer is guaranteed the API will keep working until then. The changes in Microsoft DirectX* 10 applications are:

- 1. Check for WM DISPLAYCHANGE windows message in the message handler.
- 2. Query if the graphics adapter has changed using the Windows* API's EnumDisplaySettings() or EnumDisplayDevices().
- 3. If yes, then save off the current context including state and draw information in the application and then:
 - a. Recreate the Microsoft DirectX* device.
 - b. Restore the context.
 - Continue rendering from the last scene rendered before the WM DISPLAYCHANGE message occurred.

3.10 Some suggestions, tips & tricks from the field

The items in this section are based on the observations of Intel engineers with code from developers with different levels of experiences. These are collected here as a checklist for reference for developers. Some of the items are generic to all graphics platforms.

3.10.1 Device Capabilities

Intel® microarchitecture codename Sandy Bridge supports DirectX* functionality up to and including full D3D10.1 support.

If you encounter a Direct3D feature that does not work on the latest drivers on this device, please contact your Intel Account Manager. Intel will investigate these issues for future drivers and should be able to suggest workarounds.

How to maximize graphics performance on Intel® Integrated Graphics



3.10.2 DirectX* 9 Extensions

Several hardware vendors support their own extensions to the DirectX* 9 specifications through specific texture formats and render paths that are not part of Microsoft's official DirectX* 9 specifications.

To ensure maximum compatibility with these extensions, Intel now supports many of those extensions. A list of those, current as of the release of this Guide, is given in section 3.6 Advanced DirectX* 9 Capabilities (page 24).

If there are additional extensions that you believe are useful, (or have OpenGL* extensions that you require) please let your Intel Account Manager know.

3.10.3 Revisit Assumptions on Performance

Intel® Processor Graphics is continually increasing functionality and performance. As well as the addition of full D3D10.1 support and increased capabilities previously mentioned, the performance profile has been improved significantly for this platform. As such, it is advised to remove previous restrictions and scale your title to match this increased performance and functionality.

Should you see unexpected issues, please follow these steps:

- Verify you are running the latest drivers. This platform is evolving, so there will be frequent driver updates. Check for updates at http://www.intel.com and if you are an Intel software partner, at http://platformsw.intel.com.
 - Intel graphics drivers follow a naming convention that use four field numbers for example, 15.21.8.2279. The number in the last field 2279 in the example increases sequentially with each driver release. So a driver with a higher number there is more recent. You always want to have the most recent driver installed on your system.
- 2. If you suspect that it is a functionality bug, try to recreate the bug with the reference rasterizer.
- 3. Look for easily fixed hotspots using the Intel® Graphics Performance Analyzers (Intel® GPA). Talk to your Intel Account Manager if you do not already have access to this tool.
- 4. If the above steps do not resolve the issue, or you need additional help determining the root cause, please contact us see the section Further Help Beyond this Guide in this document. Intel engineers are available to help enable your title to run effectively on our platforms. This enabling can potentially include (but is not limited to):
 - a. Training on using Intel GPA to get the best results for your title.
 - b. In-depth performance analysis of your code running on our platform, with specific feedback on optimization opportunities.
 - c. Championing the resolution of your issues within Intel, such as helping resolve or workaround driver issues, addressing tool issues, etc.

30 Document Number: 321371-002US



3.10.4 Avoid Writing to Unlocked Buffers

We have found multiple cases of corruption which were caused by the writes to unlocked buffers. Typically the title first locked the buffer, then unlocked the buffer and then attempted to write to the buffer that is already unlocked. This *sometimes* works because the system might have not moved the buffer and has not dispatched the rendering commands to the graphics hardware. This is inconsistent with Microsoft DirectX* specifications and it is not safe for applications to expect it to work consistently.

Avoid writing to buffers that have not been locked. Some drivers on other hardware platforms are more forgiving of this approach, and may handle it gracefully. The Intel driver makes optimizations based on the specified behavior of the API, so the behavior of this platform will be undefined in this case.

Always test your application with the DirectX* debug runtime enabled. The debug runtime will catch scenarios like this and help you avoid problems should driver behavior change in the future.

3.10.5 Avoid Tight Polling on Queries

Tight polling on event/occlusion queries degrades performance by interfering with the graphics subsystem's turbo mode operation.

Allow for queries to work asynchronously and avoid waiting on the query response immediately after sending the query. Issue queries as early as possible in the frame, and issue their dependent draws as late as possible. If the query result is not available at the time the draw is to be submitted just issue the draw. Any additional delay at this point will cause a pipeline stall.



4 Appendix: Sample Code for Identifying Intel® Processor Graphics and Specifying Graphics Presets

The following sample code and configuration file demonstrates how to identify Intel® Processor Graphics versions and set fidelity levels for older generations (Low) to the most recent (Medium) based on the requested D3D*_FEATURE_LEVEL.

Example Source Code:

```
// Copyright 2010 Intel Corporation
// All Rights Reserved
/// Permission is granted to use, copy, distribute and prepare derivative works of this // software for any purpose and without fee, provided, that the above copyright notice
// and this statement appear in all copies. Intel makes no representations about the // suitability of this software for any purpose. THIS SOFTWARE IS PROVIDED ""AS IS."" // INTEL SPECIFICALLY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, AND ALL LIABILITY,
// INCLUDING CONSEQUENTIAL AND OTHER INDIRECT DAMAGES, FOR THE USE OF THIS SOFTWARE, // INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, AND INCLUDING THE
// WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Intel does not
// assume any responsibility for any errors which may appear in this software nor any // responsibility to update it.
// DeviceId.cpp : Implements the Graphics Device detection and graphics settings
                         configuration functions.
#include <stdio.h>
#include <D3DCommon.h>
#include <DXGI.h>
#include <D3D9.h>
static const int FIRST GFX ADAPTER = 0;
// Define settings to reflect Fidelity abstraction levels you need
typedef enum {
                     NotCompatible, // Found Graphics is not compatible with the app
                     Medium,
                     High,
                     Undefined
                                            // No predefined setting found in cfg file.
                                             // Use a default level for unknown video cards.
                  PresetLevel:
 * getGraphicsDeviceID
         Function to get the primary graphics device's Vendor ID and Device ID, either through the new DXGI interface or through the older D3D9 interfaces.
```

32 Document Number: 321371-002US



```
bool getGraphicsDeviceID( unsigned int& VendorId, unsigned int& DeviceId )
    bool retVal = false;
   bool bHasWDDMDriver = false;
    HMODULE hD3D9 = LoadLibrary( L"d3d9.d11" );
    if ( hD3D9 == NULL )
        return false;
    * Try to create IDirect3D9Ex interface (also known as a DX9L interface).
     ^{\star} This interface can only be created if the driver is a WDDM driver.
    // Define a function pointer to the Direct3DCreate9Ex function.
    typedef HRESULT (WINAPI *LPDIRECT3DCREATE9EX) ( UINT, void **);
    // Obtain the address of the Direct3DCreate9Ex function.
    LPDIRECT3DCREATE9EX pD3D9Create9Ex = NULL;
   pD3D9Create9Ex = (LPDIRECT3DCREATE9EX) GetProcAddress( hD3D9, "Direct3DCreate9Ex" );
   bHasWDDMDriver = (pD3D9Create9Ex != NULL);
    if( bHasWDDMDriver )
        // Has WDDM Driver (Vista, and later)
        HMODULE hDXGI = NULL;
        hDXGI = LoadLibrary( L"dxgi.dll" );
        // DXGI libs should really be present when WDDM driver present.
        if ( hDXGI )
            // Define a function pointer to the CreateDXGIFactoryl function.
            typedef HRESULT (WINAPI *LPCREATEDXGIFACTORY) (REFIID riid, void **ppFactory);
             / Obtain the address of the CreateDXGIFactorv1 function.
            LPCREATEDXGIFACTORY pCreateDXGIFactory = NULL;
            pCreateDXGIFactory = (LPCREATEDXGIFACTORY) GetProcAddress( hDXGI,
                                                                        "CreateDXGIFactory" );
            if ( pCreateDXGIFactory )
                // Got the function hook from the DLL
                // Create an IDXGIFactory object.
                IDXGIFactory * pFactory;
                // Enumerate adapters.
                    // Code here only gets the info for the first adapter.
                    // If secondary or multiple Gfx adapters will be used, // the code needs to be modified to accomodate that.
                    IDXGIAdapter *pAdapter;
                    if ( SUCCEEDED( pFactory->EnumAdapters( FIRST GFX ADAPTER,
                                                             &pAdapter ) ) )
                        DXGI ADAPTER DESC adapterDesc;
                        pAdapter->GetDesc( &adapterDesc );
                        // Extract Vendor and Device ID information from adapter descriptor
                        VendorId = adapterDesc.VendorId;
DeviceId = adapterDesc.DeviceId;
                        pAdapter->Release();
                        retVal = true;
            FreeLibrary( hDXGI );
    else
```

How to maximize graphics performance on Intel® Integrated Graphics



```
^{\star} Does NOT have WDDM Driver. We must be on XP.
          * Let's try using the Direct3DCreate9 function (instead of DXGI)
         // Define a function pointer to the Direct3DCreate9 function.
        typedef IDirect3D9* (WINAPI *LPDIRECT3DCREATE9) ( UINT );
         // Obtain the address of the Direct3DCreate9 function.
        LPDIRECT3DCREATE9 pD3D9Create9 = NULL;
        pD3D9Create9 = (LPDIRECT3DCREATE9) GetProcAddress( hD3D9, "Direct3DCreate9");
         if( pD3D9Create9 )
               / Initialize the D3D9 interface
             LPDIRECT3D9 pD3D = NULL;
             if ( (pD3D = (*pD3D9Create9) (D3D SDK VERSION)) != NULL )
                  D3DADAPTER IDENTIFIER9
                                             adapterDesc;
                  // Enumerate adapters. Code here only gets the info for the first adapter. if ( pD3D->GetAdapterIdentifier( FIRST_GFX_ADAPTER, 0,
                                                       &adapterDesc ) == D3D_OK )
                      VendorId = adapterDesc.VendorId;
DeviceId = adapterDesc.DeviceId;
                      retVal = true;
                 pD3D->Release();
        }
    FreeLibrary( hD3D9 );
    return retVal;
 \star getDefaultFidelityPresets
       Function to find / set the default fidelity preset level, based on the type
       of graphics adapter present.
       The guidelines for graphics preset levels for Intel devices is a generic one
       based on our observations with various contemporary games. You would have to change it if your game already plays well on the older hardware even at high
PresetLevel getDefaultFidelityPresets( void )
    unsigned int VendorId, DeviceId;
    PresetLevel presets = Undefined;
    if ( !getGraphicsDeviceID ( VendorId, DeviceId ) )
        return NotCompatible;
    FILE *fp = NULL;
    switch( VendorId )
        case 0x8086:
             fopen_s ( &fp, "IntelGfx.cfg", "r" );
           // Add cases to handle other graphics vendors...
        default:
             break;
```

34 Document Number: 321371-002US



```
if (fp)
      char line[100];
      char *context = NULL;
      char *szVendorId = NULL;
char *szDeviceId = NULL;
      char *szPresetLevel = NULL;
      while ( fgets ( line, countof(line), fp ) ) // read one line at a time till EOF
             // Parse and remove the comment part of any line int i; for( i = 0; line[i] && line[i]!=';'; ++i ); line[i] = '\0';
            // Try to extract VendorId, DeviceId and recommended Default Preset Level szVendorId = strtok_s( line, ",\n", &context ); szDeviceId = strtok_s( NULL, ",\n", &context ); szPresetLevel = strtok s( NULL, ",\n", &context );
             ( szPresetLevel == NULL ) )
                   continue; // blank or improper line in cfg file - skip to next line
            unsigned int vId, dId;
sscanf_s( szVendorId, "%x", &vId );
sscanf_s( szDeviceId, "%x", &dId );
            // If current graphics device is found in the cfg file, use the
// pre-configured default Graphics Presets setting.
if( ( vId == VendorId ) && ( dId == DeviceId ) )
                   // Found the device
                   char s[10];
                   sscanf_s( szPresetLevel, "%s", s, _countof(s) );
                   if (! stricmp(s, "Low"))
                         presets = Low;
                   else if (!_stricmp(s, "Medium"))
    presets = Medium;
else if (!_stricmp(s, "High"))
                        presets = High;
                   else
                        presets = NotCompatible;
                   break; // Done reading file.
      fclose( fp ); // Close open file handle
// If the current graphics device was not listed in any of the config // files, or if config file not found, use Low settings as default. if ( presets == Undefined )
      presets = Low;
return presets;
```

Example Configuration File:

```
; Intel Graphics Preset Levels
;
; Format:
; VendorIDHex, DeviceIDHex, CapabilityEnum ; Commented name of cards
;

0x8086, 0x2582, Low ; SM2 ; Intel(R) 82915G/GV/910GL Express Chipset Family
0x8086, 0x2782, Low ; SM2 ; Intel(R) 82915G/GV/910GL Express Chipset Family
0x8086, 0x2592, Low ; SM2 ; Mobile Intel(R) 82915GM/GMS, 910GML Express Chipset Family
```

How to maximize graphics performance on Intel® Integrated Graphics



Intel® Processor Graphics DirectX* Developer's Guide

```
        0x8086,
        0x2792,
        Low
        ;
        SM2

        0x8086,
        0x2772,
        Low
        ;
        SM2

        0x8086,
        0x2776,
        Low
        ;
        SM2

        0x8086,
        0x27A2,
        Low
        ;
        SM2

        0x8086,
        0x27A6,
        Low
        ;
        SM2

        0x8086,
        0x2972,
        Low
        ;
        SM2

        0x8086,
        0x2973,
        Low
        ;
        SM2

        0x8086,
        0x2993,
        Low
        ;
        SM2

        0x8086,
        0x2982,
        Low
        ;
        SM2

        0x8086,
        0x29B3,
        Low
        ;
        SM2

        0x8086,
        0x29B3,
        Low
        ;
        SM2

        0x8086,
        0x29C2,
        Low
        ;
        SM2

        0x8086,
        0x29C3,
        Low
        ;
        SM2

        0x8086,
        0x29D3,
        Low
        ;
        SM2

        0x8086,
        0x29D3,
        Low
        ;
        SM2

   0x8086, 0x2792, Low ; SM2 ; Mobile Intel(R) 82915GM/GMS, 910GML Express Chipset Family
                                                                                                          ; Intel(R) 82945G Express Chipset Family ; Intel(R) 82945G Express Chipset Family
                                                                                                        ; Intel(R) 82945G Express Chipset Family; Mobile Intel(R) 945GM Express Chipset Family; Mobile Intel(R) 945GM Express Chipset Family; Intel(R) 946GZ Express Chipset Family; Intel(R) 946GZ Express Chipset Family; Intel(R) 946GZ Express Chipset Family; Intel(R) Q965/Q963 Express Chipset Family; Intel(R) Q35 Express Chipset Family; Intel(R) Q35 Express Chipset Family; Intel(R) Q35 Express Chipset Family; Intel(R) Q33/G31 Express Chipset Family
                                                                                                         ; Intel(R) G33/G31 Express Chipset Family
; Intel(R) G33/G31 Express Chipset Family
; Intel(R) Q33 Express Chipset Family
; Intel(R) Q33 Express Chipset Family
 0x8086, 0xA001, Low ; SM2
0x8086, 0xA002, Low ; SM2
0x8086, 0xA011, Low ; SM2
0x8086, 0xA012, Low ; SM2
                                                                                                           ; Intel(R) NetTop Atom D410 (GMA 3150)
; Intel(R) NetTop Atom D510 (GMA 3150)
; Intel(R) NetBook Atom N4x0 (GMA 3150)
                                                                                                           ; Intel(R) NetBook Atom N4x0 (GMA 3150)
 0x8086, 0x29A2, Low ; SM3
0x8086, 0x29A3, Low ; SM3
                                                                                                            ; Intel(R) G965 Express Chipset Family
                                                                                                            ; Intel(R) G965 Express Chipset Family
 0x8086, 0x8108, Low ; SM3
0x8086, 0x8109, Low ; SM3
                                                                                                            ; Intel(R) GMA 500 (Poulsbo) on MID platforms
                                                                                                              ; Intel(R) GMA 500 (Poulsbo) on MID platforms
0x8086, 0x2982, Low ; SM4
0x8086, 0x2983, Low ; SM4
0x8086, 0x2A02, Low ; SM4
0x8086, 0x2A02, Low ; SM4
0x8086, 0x2A12, Low ; SM4
0x8086, 0x2A12, Low ; SM4
0x8086, 0x2A13, Low ; SM4
0x8086, 0x2A42, Low ; SM4
0x8086, 0x2A42, Low ; SM4
0x8086, 0x2E02, Low ; SM4
0x8086, 0x2E02, Low ; SM4
0x8086, 0x2E23, Low ; SM4
0x8086, 0x2E23, Low ; SM4
0x8086, 0x2E13, Low ; SM4
0x8086, 0x2E13, Low ; SM4
0x8086, 0x2E13, Low ; SM4
0x8086, 0x2E32, Low ; SM4
0x8086, 0x2E32, Low ; SM4
0x8086, 0x2E32, Low ; SM4
0x8086, 0x2E33, Low ; SM4
0x8086, 0x2E343, Low ; SM4
0x8086, 0x2E43, Low ; SM4
0x8086, 0x2E92, Low ; SM4
0x8086, 0x2E93, Low ; SM4
                                                                         ; SM4
 0x8086, 0x2982, Low
0x8086, 0x2983, Low
                                                                                                            ; Intel(R) G35 Express Chipset Family
                                                                                                              ; Intel(R) G35 Express Chipset Family
                                                                                                            ; Mobile Intel(R) 965 Express Chipset Family
                                                                                                              ; Mobile Intel(R) 965 Express Chipset Family
                                                                                                          ; Mobile Intel(R) 965 Express Chipset Family
; Mobile Intel(R) 965 Express Chipset Family
; Mobile Intel(R) 965 Express Chipset Family
; Mobile Intel(R) 4 Series Express Chipset Family
; Mobile Intel(R) 4 Series Express Chipset Family
; Intel(R) 4 Series Express Chipset
                                                                                                         ; Intel(R) 4 Series Express Chipset
; Intel(R) 4 Series Express Chipset
; Intel(R) 645/643 Express Chipset
; Intel(R) 645/643 Express Chipset
; Intel(R) Q45/Q43 Express Chipset
; Intel(R) Q45/Q43 Express Chipset
; Intel(R) 641 Express Chipset
; Intel(R) 641 Express Chipset
; Intel(R) B43 Express Chipset
; Intel(R) B43 Express Chipset
                                                                                                          ; Intel(R) B43 Express Chipset
; Intel(R) B43 Express Chipset
                                                                                                         ; Intel(R) B43 Express Chipset
; Intel(R) Processor Graphics - Core i3/i5/i7 Mobile
   Processors
  0x8086, 0x0042, Low ; SM4
                                                                                                          ; Intel(R) Processor Graphics - Core i3/i5 + Pentium G9650
  Processors
 0x8086, 0x0106, Low ; SM4.1 ; Mobile SandyBridge Processor GRAPHICS 2000 0x8086, 0x0102, Low ; SM4.1 ; Desktop SandyBridge Processor GRAPHICS 2000 0x8086, 0x010A, Low ; SM4.1 ; SandyBridge Server
  0x8086, 0x0112, Medium ; SM4.1 ; Desktop SandyBridge Processor GRAPHICS 3000
 0x8086, 0x0112, Medium; SM4.1; Desktop SandyBridge Processor GRAPHICS 3000
0x8086, 0x0116, Medium; SM4.1; Mobile SandyBridge Processor GRAPHICS 3000
  0x8086, 0x0126, Medium; SM4.1; Mobile SandyBrdige Processor GRAPHICS 3000
```

36 Document Number: 321371-002US



5 Support

- Intel's processor graphics chipset development community forum:
 - http://software.intel.com/en-us/forums/developing-software-for-visual-computing/
- Game programming resources:
 - http://software.intel.com/en-us/visual-computing/
- Intel® Software Network:
 - http://software.intel.com/en-us/
- Intel® Software Partner Program:
 - http://www.intel.com/software/partner/visualcomputing/
- Intel® Visual Adrenaline graphics and gaming campaign:
 - http://www.intel.com/software/visualadrenaline/
- Intel® Graphics Performance Analyzers (Intel® GPA):
 - http://software.intel.com/en-us/articles/intel-gpa/
- Intel® Parallel Studio:
 - http://software.intel.com/en-us/articles/intel-parallel-studio-home/
- Intel® VTune™ Amplifier XE (Formerly Intel® VTune™ Performance Analyzer):

http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/



6 References

- [1] "Copying and Accessing Resource Data (Direct3D 10)". Direct3D Programming Guide. Microsoft DirectX* SDK (November 2008).
- [2] "DirectX* Constants Optimizations for Intel processor graphics". Intel Software Network, Intel: http://software.intel.com/en-us/articles/directx-constants-optimizations-for-intel-integrated-graphics/.

38 Document Number: 321371-002US



THE RISE OF MOBILE GAMING ON ANDROID: QUALCOMM® SNAPDRAGON™ TECHNOLOGY LEADERSHIP

© 2014 Qualcomm Technologies, Inc. All Rights Reserved.

ATI 2092 LG v. ATI IPR2015-00326

Qualcomm Technologies, Inc.

Qualcomm, Snapdragon, Adreno, FlexRender, Trepn, Vuforia, Brew and Hexagon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. TruPalette, ecoPix, Krait and Smart Terrain are trademarks of Qualcomm Incorporated. All Qualcomm Incorporated trademarks are used with permission. AllJoyn is a trademark of Qualcomm Innovation Center, Inc., registered in the United States and other countries, used with permission. Other products and brand names may be trademarks or registered trademarks of their respective owners.

Qualcomm Snapdragon, Qualcomm Adreno, Qualcomm TruPalette, Qualcomm ecoPix, FlexRender, Trepn, Brew, Hexagon and Krait are products of Qualcomm Technologies, Inc. Qualcomm Vuforia is a product of Qualcomm Connected Experiences, Inc. Smart Terrain is a feature of the Qualcomm Vuforia SDK. AllJoyn open source project is hosted by the Allseen Alliance.

Qualcomm Technologies, Inc. 5775 Morehouse Drive San Diego, CA 92121 U.S.A.

© 2014 Qualcomm Technologies, Inc. All Rights Reserved.

Table of Contents

1 Executive summary			
2	Th	e rise of mobile gaming on Android	1
3	lm	mersive mobile gaming experiences at low power	2
	3.1	Heterogeneous computing: specialized engines designed for efficient processing4	
	3.2	Efficient SoC architecture: smart management of system resources	
4	Pre	oviding a consistent development platform at scale	. 10
	4.1	Android gaming on Snapdragon: an excellent opportunity10	
	4.2	Scalable architecture: consistent gaming experiences across various tiers	
5	Un	locking the potential of Snapdragon processors for developers	.12
	5.1	Supporting developers with comprehensive tools	
	5.2	Supporting the developer community	
	5.3	Next-generation gaming experiences	
6	Co	nclusion	. 18

1 Executive summary

Mobile gaming is the fastest growing segment in the game industry. Today's consumers want immersive, connected gaming with all-day battery life that provides visually stunning graphics and high-fidelity audio. Android is helping fulfill that desire. With a large user base, Android devices provide a growing opportunity for game developers to generate revenue on a global scale.

The rapid iteration and fast innovation of Android have created a thriving ecosystem with numerous custom-designed form factors across device tiers. However, there are three challenges to taking Android gaming to the next level, including:

- How to create immersive gaming experiences within the power and thermal constraints of mobile devices?
- How to develop a game that addresses a sizeable portion of the mobile gaming segment with minimal code variations?
- How to take advantage of the hardware's capabilities without sacrificing time-to-market?

Qualcomm Technologies, Inc. (QTI) was the number one provider of Android smartphone application processor (AP) shipments in 2013.² Its Qualcomm® Snapdragon™ processors³ are fully integrated system on a chip (SoC) solutions designed with mobile in mind, handling everything from the most advanced, console-quality mobile games to the most popular casual games, all while delivering long battery life. This paper will describe the rise of mobile gaming on Android and how QTI is meeting the three challenges by:

- Creating low-power mobile processors by taking a heterogeneous computing (HC) approach and designing an efficient SoC architecture.
- Providing a consistent development platform across tiers due to its scalable architecture.
- Offering tools, support, and advanced technologies to easily unlock the full potential of Snapdragon processors for developers.

2 The rise of mobile gaming on Android

Mobile gaming is the fastest growing segment in the game industry, with a 30% compound annual revenue growth rate projected from 2013 to 2015.¹

Android burst onto the scene in 2008. The *open* nature and rapid iteration of the platform, combined with improving hardware, and a growing installed base of devices worldwide, has created excellent opportunities for the mobile gaming ecosystem.

Source: Gartner, October 2013, "Forecast Video Game Ecosystem Worldwide"

Source: Strategy Analytics, April 2014

Snapdragon processors extend across four product tiers: Snapdragon 800, Snapdragon 600, Snapdragon 400, and Snapdragon 200.

Through standard application program interfaces (APIs), Android exposes key platform capabilities to developers, such as connectivity, sensors, and graphics rendering. Android also has a history of quickly adopting the latest graphics standards, like OpenGL ES. The consistent release of new APIs helps limit fragmentation, improve ease of use, and decrease time-to market.

To take advantage of increasing consumer engagement on the Google Play app store, developers are now creating games that utilize the latest Android capabilities (e.g., 3D graphics). As a result, new game brands have been introduced (e.g., *Real Soccer, Asphalt, Real Racing*, and *Modern Combat*). Debuting in the smartphone, Android is now extending to other form factors, including tablets, handheld gaming devices, set-top-boxes, TVs, and more.

Total mobile gaming revenues (for all platforms) are projected to grow from \$13 billion in 2013 to \$22 billion in 2015⁴. Within this category, the Android platform provides a large and growing opportunity for developers. On the device side, 78.4% of smartphones shipped last year were Android, outselling devices based on the next leading smartphone mobile OS by almost 5x.⁵ Android device shipments (smartphones and tablets) are projected to exceed one billion units in 2014.⁶ On the apps side, 75% of Android users play games⁷, accounting for 90% of the app-generated revenue on Google Play⁸.

3 Immersive mobile gaming experiences at low power

Immersive gaming requires more than just console-quality graphics. Other elements include cinemaquality sound, realistic effects, instantaneous response times (low latency), and the ability to play anywhere. Figure 1 highlights the key components that contribute to the overall gaming experience.

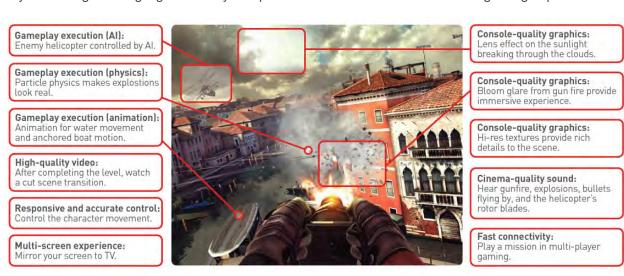


Figure 1: Example of key gaming components in Modern Combat 5: Blackout by Gameloft

Gartner, October 2013, "Forecast Video Game Ecosystem Worldwide"

Gartner, "Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013"

Gartner, "Gartner Says Worldwide Traditional PC, Tablet, Ultramobile and Mobile Phone Shipments On Pace to Grow 7.6% in 2014"

Kris Holt, "Google Adds Some Serious Oomph to Play Games"

eMarketer, "Game Apps Are No. 1 for Amazon, Apple and Google"

Today's mobile graphics are approaching the visual quality of consoles and PCs with high-end graphics cards.





Figure 2: Mobile (left) vs. PC (right) rendering for Epic Unreal Engine 4

Consumers expect console-quality graphics and long battery life when playing any mobile game, from simple casual games like *Candy Crush* to more immersive games like *Modern Combat 5: Blackout*. However, enabling advanced mobile gaming experiences for sustained periods is challenging. Consoles and PCs have the luxury of being able to use fans (and large heat sinks) while drawing triple-digit wattages from an external power outlet. Mobile devices, on the other hand, are *passively* cooled, powered by a battery, and limited to single-digit wattages in order to meet the power and thermal constraints of the thin form factor.

Through its Snapdragon processors, QTI is addressing the performance and power challenge by:

- Taking a power-optimized, heterogeneous approach to mobile computing.
- Analyzing performance bottlenecks and designing an *efficient SoC architecture* that improves each generation.



Figure 3: Asphalt 8: Airborne by Gameloft

"Qualcomm® Snapdragon™ processors really help enable us to create an immersive experience for the gamer..."

Sylvain Baudry Business Development Director Gameloft

3.1 Heterogeneous computing: specialized engines designed for efficient processing

QTI has a long history of taking a heterogeneous computing approach. This approach intelligently utilizes specialized engines, such as the GPU, DSP, and display engine, to support new immersive experiences, while helping to maximize battery life and thermal efficiency.

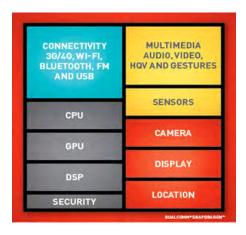


Figure 4: Snapdragon processing engines

As mentioned in Section 3, games are comprised of multiple components. These components may be implemented as one or more tasks. To support the optimal gaming experience at the lowest power and thermal levels, each task should be designed to run on the most appropriate engine. Figure 5 is an example of this.

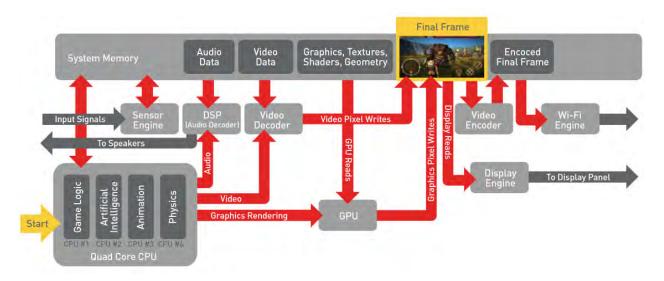


Figure 5: High-level view of gaming tasks being appropriately distributed to specialized engines

The rest of this section provides examples of how Snapdragon processors are designed to run key gaming tasks on the most appropriate processing engines.

Controlling gameplay execution — **keeping everything in sync:** The CPU dispatches work to the appropriate specialized engine and processes individual functions within the game as separate tasks. For example AI, animation, physics, and gameplay are processed on the CPU.

Functions like physics and AI require high-precision math, which can be efficiently processed on the VeNum floating point units of QTI's custom-designed Krait™ CPU. In addition, because the tasks are independent, they can be written as separate threads and run on separate CPUs for optimal performance and latency.

QTI has been innovating its power management and scheduling software for years to provide the different amounts of processing that various tasks require. For example, Krait CPUs have long employed asynchronous symmetric multi-processing (aSMP) so that optimized performance is delivered per CPU core, thus saving power.

Console-quality graphics — advanced features and graphics rendering (at low power): When most people think of modern mobile gaming, it is the console-quality graphics that immediately come to mind. Rendering graphics at low power and at high frame rates requires a specialized engine, the GPU. QTI's custom-designed Qualcomm® Adreno™ GPUs are a family of low power, fully programmable GPUs that are designed for optimal mobile gaming performance.

Adreno GPUs provide comprehensive support for the latest graphics APIs. The Khronos OpenGL ES specifications define the primary graphics APIs for Android gaming. QTI works closely with the Khronos group to help define the standards, and the Adreno GPU is optimized to support OpenGL ES (from the silicon to the drivers and software stack).

Adreno GPUs have a long history of graphics technology leadership. They support many design innovations for efficient graphics processing, including a flexible, power-efficient unified shader architecture, which is designed to support dynamic resource allocation (for optimal shader processing).

"Epic now has brought Unreal Engine 4 to Android with the Snapdragon 800 and 805 chipsets from Qualcomm Technologies. Recently we worked with Qualcomm [Technologies] to elevate graphics to the next level on the ... Adreno GPU hardware, which delivers some of the most power-efficient unified shader capabilities we've seen yet for Android smartphones and tablets." — Niklas "Smedis" Smedberg, Epic Games

The Adreno GPU also supports dynamic <u>FlexRender™</u> technology, which is designed to intelligently choose between immediate/direct mode and deferred/tile-based rendering, to render various tasks (such as user interface and gameplay) in a more efficient manner.

Custom designing the Adreno GPUs allows QTI to evolve the architecture in a timely manner for emerging mobile use cases, such as new gaming features and APIs. For example, the newest Adreno 4x series GPUs are designed to the DirectX11FL_11_2 specification, the same graphics specification as today's high-end desktop graphics cards and the latest gaming consoles.

To support next-gen features (beyond the newly announced Khronos OpenGL ES 3.1 APIs), Adreno 4x series GPUs extended their unified shader architecture by adding several completely new shaders (e.g., geometry and tessellation).

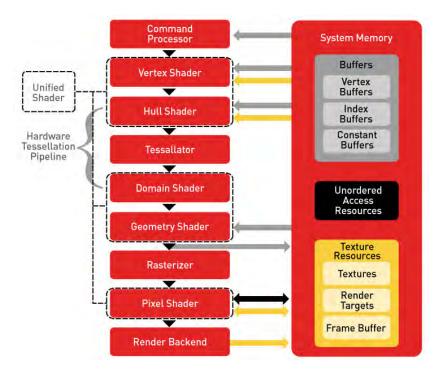


Figure 6: Adreno 4x series GPU's DirectX11 FL_12 based 3D hardware pipeline

The Adreno 420 GPU is one of the first commercial mobile GPUs to support dynamic hardware tessellation. Dynamic hardware tessellation is designed to help developers to provide greater detail for more visually realistic scenes in a manner that requires less memory bandwidth, lower power consumption, and lower overall memory footprint for the application.

Traditionally, in order for most 3D games to be visually immersive, the programmer must include a substantial amount of geometry detail per object. The denser the geometric mesh used to create an object in the scene (human or monster for example) of a game, the smoother and more realistic its surfaces will look. In traditional mobile GPUs, these additional geometry assets (required for improved visual realism) need to be stored in memory as part of the game binary package. These assets have to be copied or written by the CPU to the system memory and eventually have to be read into the GPU for further processing. These additional read and write operations could increase the memory bandwidth usage and power consumption to unsustainable levels (for console-quality games) on mobile devices.

Dynamic hardware tessellation helps solve this problem by allowing the GPU to generate additional geometry on-chip, without requiring additional data transfers from off-chip, system memory. Dynamic hardware tessellation can help significantly reduce bandwidth and power consumption. Another advantage is that the developer doesn't have to author and store these additional geometry details into their game package, which significantly reduces the memory footprint and the overall binary size of the game.

The image below shows the additional detail that tessellation provides to both the wireframe and the final image rendered. For this simple "Hornet" graphics scene, hardware tessellation delivers a bandwidth savings of ~360MB/s, and a memory footprint savings of ~20MB⁹. For larger games, the savings on memory footprint could be in GBs.





Figure 7: Tessellation (OFF: left, ON: right) provides more realistic and detailed graphics

By providing more detail, tessellation is designed to help significantly improve the gaming experience, not only for high resolution content, but also for low resolution content that needs to be upscaled.

The Adreno 4x series GPUs are a great example of QTI's commitment to bringing console-quality gaming up to 4K Ultra HD (4K) resolution to mobile devices.

"[Adreno 420 is] the most aggressive move in mobile graphics by any company, to add all the shader types, and HW tessellation, on top of what they did in Subdiv for Motorola, shows Qualcomm [Technologies] as the most committed mobile graphics supplier today. It really is bringing console-class graphics to mobile devices." — Jon Peddie, Jon Peddie Research.

Multi-screen experiences — **high fidelity on your screens:** Playing a game on a high-quality, high-resolution screen significantly enhances the gaming experience. The display engine enhances images, composites multiple images, and supports high resolution (both on-device and on external displays) of up to 4K.

For image enhancement, the display engine utilizes QTI's Qualcomm® TruPalette™ and Qualcomm® EcoPix™ feature sets, which include high quality post-processing algorithms for superior picture quality, including picture adjustment, color enhancement, contrast enhancement, scaling, sharpening, and power efficiency. For example, using the ecoPix sunlight visibility improvement technology, the display engine can use information from the device's light sensor to enhance the rendered game content to make it much more visible in bright conditions (e.g., outdoors on a sunny day) through tone correction.

Numbers based on QTI internal testing

Jon Peddie, "Qualcomm Moves to 4K with Snapdragon 805"



Figure 8: Snapdragon display engine TruPalette and EcoPix feature sets

Device-to-device connectivity further elevates the mobile gaming experience. External displays, such as a TV, are driven by either a wired (e.g., HDMI) or wireless connection (e.g., Miracast). It is the task of the display engine to efficiently process both types of connections. With the wireless display feature, you can send the contents of your mobile device screen to your smart TV screen. QTI supports wireless display standards, like Miracast, and is adding wireless display support to an increasing number of Android devices.

Cinema-quality sound — efficient audio processing in sync with the graphics: Just as sound is a huge part of the movie-going experience, it is also important in the world of immersive mobile gaming. Accordingly, QTI provides a comprehensive audio solution, including hardware and software, which offers high-fidelity audio and advanced features like 24-bit/96kHz play back and cinema-quality 7.1 surround sound audio.

The processing required to efficiently support these computationally complex audio features is primarily handled by QTI's Hexagon™ DSP, which is custom designed for heavy signal processing tasks, like audio. The real-time processing capabilities of QTI's Hexagon DSP are designed to keep the audio in sync with the graphics rendering. QTI has also worked with industry leaders, such as DTS and Dolby, to provide an optimal audio experience on Snapdragon processors, including support for headphone surround virtualization.

High-quality video — **specialized video engine:** Some games incorporate pre-rendered cut scenes to enhance the overall gaming experience. To save memory, these scenes are compressed into encoded video. To playback video, Snapdragon processors are designed to use a specialized video engine to decompress encoded videos at low power. For higher resolutions and more complex codecs, a specialized engine becomes even more important.

Figure 9 shows that running H.265 HEVC decode on a specialized video engine versus a CPU reduces power consumption, while still meeting the frame rate requirements¹¹.

¹¹ Estimated numbers based on QTI internal testing

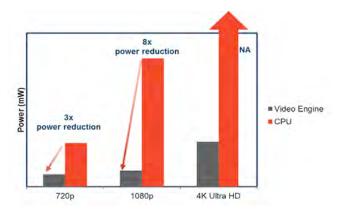


Figure 9: Power reduction by running HEVC decode on specialized video engine versus the CPU

Fast connectivity — **multi-player gaming:** Connected gaming experiences, like massively multiplayer online (MMO) games, require fast and reliable connectivity solutions with low latency where the difference between winning and losing often comes down to milliseconds. To achieve that, Snapdragon processors include connectivity technologies, such as 802.11ac Wi-Fi and 3G/4G LTE.

QTI has a long history of being an industry leader in advanced connectivity solutions. Its latest Snapdragon 810 processor continues the tradition, integrating a Cat 6 LTE Advanced multimode modem that is designed to support reliable communication at speeds of up to 300 Mbps. QTI optimizes the software stack for connectivity to achieve low client-server ping times on Snapdragon processors, so gamers can worry less about a slow connection and focus more on enjoying the game.

Responsive and accurate control — supporting multiple input methods: A good gaming experience requires an input method that gives precise control at low latency. There are multiple ways to control and interface to games, such as through a touch screen, device movement, a game controller, and gestures. Snapdragon processors have been designed to reduce latency and provide accurate control for these methods. For example, device movement generally is determined by processing data from motion sensors, such as the accelerometer or gyroscope. Sensor processing requires intensive signal processing, control processing, and real-time processing. QTI's specialized sensor engine, which excels at handling these tasks, is integrated in Snapdragon processors. As a result, the response time for device movement is faster — and at low power.

3.2 Efficient SoC architecture: smart management of system resources

An efficient SoC architecture is required in order to sustain console-quality gaming at high frame rates within the thermal and power constraints of mobile devices. As noted above, the majority of the processing engines within the SoC are running concurrently in a heterogeneous manner to efficiently process gaming tasks. Beyond heterogeneous computing, smartly managing the shared system resources, such as memory bandwidth, power budget, and thermal budget, is necessary to support sustained gameplay of a visually complex game.

Efficient memory bandwidth: While playing a game, the processing engines need to be fed data (starving a processing engine of data can lead to lower frame rates). To prevent bottlenecks, Snapdragon processors use advanced memory management techniques. The memory controller is designed to deliver high quality of service (QoS) for different throughput and latency requirements of different processing engines, while still maximizing the memory utilization. By minimizing the overhead associated with memory transactions, the memory controller helps increase memory utilization and minimizes power consumption.

Snapdragon processors are also designed to deploy smart caching mechanisms in many processing engines to help minimize the need for frequent DRAM access. For example, the Adreno GPU's tile-based architecture, which subdivides the graphics image into smaller tiles and renders them to the tile-buffer cache, helps minimize DRAM bandwidth and saves power.

Advanced power and thermal management technology: Drawing high power not only reduces battery life, it also releases excess heat. This will raise the skin temperature of the device, making it uncomfortable to hold. To reduce the system power, Snapdragon processors are designed to deploy sophisticated algorithms that manage power based on workload requirements. They support a wide range Dynamic Clock and Voltage Scaling (DCVS). DCVS dynamically varies the clock frequencies and voltages of the processing engines. For example, the Adreno 420 GPU now has more granular DCVS power control levels, so it can run most use cases at a nominal voltage state, thus saving power.

Several of the specialized engines have innovative, power-saving techniques. For example, the display engine also uses a proprietary compression scheme called Frame Buffer Compression (FBC), which compresses display data by up to 66% in a visually lossless manner—before transmission to the display panel.¹²

The Adreno 420 GPU also has an increased Z-reject rate for graphics rendering so that pixels that are not going to be visible (because they are blocked by a pixel on top) are not processed. An increased Z-reject rate means lower power per pixel—and improved performance.

4 Providing a consistent development platform at scale

To attract game developers, devices with Snapdragon processors provide an excellent opportunity with a consistent development environment.

4.1 Android gaming on Snapdragon: an excellent opportunity

As of January 2014, over 1,350 devices with Snapdragon processors had been announced or were commercially shipping. In addition, more than 525 new device designs are in development, and QTI is currently working with over 85 manufacturers worldwide. QTI was the #1 provider of Android smartphone AP shipments in 2013.¹³

Strategy Analytics, April 2014

¹² QTI internal metrics

By optimizing their games on Snapdragon processors, developers have the opportunity to reach this very large and growing installed base of Android devices with Snapdragon processors that span across various price tiers and regions.



Figure 10: Devices with Snapdragon processors

Examples include iconic flagship Android-based devices like the Nexus 5 by Google, LG G Pro2, and Samsung Galaxy Note 3. Recently launched devices, including the HTC One (M8), One+ One, LG G3, Samsung Galaxy S5, and the Sony Xperia Z2, are powered by the Snapdragon 801 processor. Amazon Kindle Fire HDX 8.9 and Fire TV, based on Fire OS, are also powered by Snapdragon processors.

Refer to the <u>smartphones</u> and <u>tablets</u> websites for the latest devices powered by Snapdragon processors. With Adreno 4x series GPUs on the verge of being launched in commercial devices, Snapdragon processors continue to provide graphics and gaming technology leadership.

According to Jon Peddie Research, "Qualcomm [Technologies] is clearly the largest SoC supplier". For Q2 2013 and as indicated in Figure 11, QTI was the single largest GPU supplier for personal devices, which includes smartphones, tablets, and dedicated handheld game consoles. ¹⁴

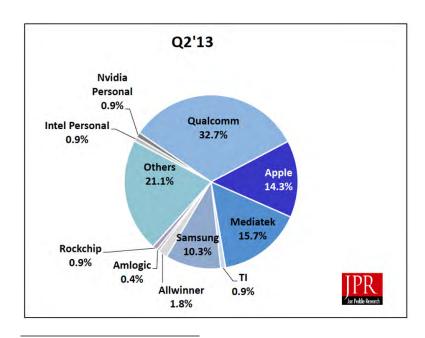


Figure 11: Share of personal systems SoC suppliers for Q2 2013

Jon Peddie Research, Oct. 2013, "Mobile Devices and the GPUs inside"

4.2 Scalable architecture: consistent gaming experiences across various tiers

Consumers expect immersive gaming experiences whether they own a premium or an entry-level mobile device. Snapdragon processors are designed to support great gaming experiences across the spectrum of device tiers. Apart from offering backwards-compatible software between different tiers, Snapdragon processors with the scalable Adreno GPU are designed to provide consistent features and APIs across tiers. Using APIs as an example, Adreno 3x series GPUs (and above) support OpenGL ES 3.0. Having common support of APIs and features ranging from entry-level to premium devices helps make it easier for developers to introduce next-generation features to mass audiences, without having to custom design for each tier.

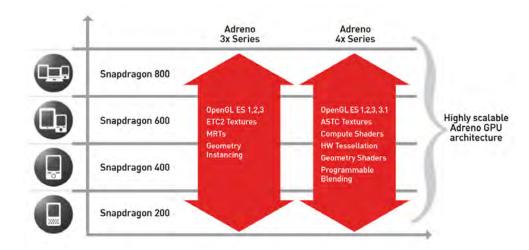


Figure 12: Snapdragon processors with the scalable Adreno GPU are designed to provide consistent features across tiers

Snapdragon processors are software compatible, helping both OEMs and developers to efficiently invest and develop across multiple device types and tiers. With a consistent software stack, including graphics drivers, devices with Snapdragon processors are designed to provide a consistent and optimized gaming platform across various tiers.

5 Unlocking the potential of Snapdragon processors for developers

QTI and its affiliates have a long history of technology leadership in supporting the mobile gaming ecosystem, dating back to 2001 with the Brew™ platform. This breakthrough development platform allowed native C/C++ games to be written "closer to the metal," supporting higher-quality and higher-performance games, and helped usher in some of the world's first 3D games in mobile. Brew was also one of the earliest monetization platforms for mobile applications, paying out more than \$3 billion to developers¹⁵.

https://www.brewmp.com/about

The introduction of 3G in the early 2000s enabled faster downloads of larger games, multi-player gaming, and more robust client-server-based game designs. In addition, Qualcomm Ventures also helped fund mobile gaming pioneers such as JAMDAT Mobile and more recently invested in companies such as Bluestacks, Gaikai, Grand Cru, Playdek, Playnery, and TabTale.

It takes more than just great hardware to advance the mobile gaming industry. To support game developers in producing immersive games, QTI provides comprehensive tools, extensive support, and advanced technologies.

5.1 Supporting developers with comprehensive tools

QTI supports developers in unlocking the performance and advanced features of Snapdragon processors by not only offering comprehensive development tools, but by also working closely with the gaming ecosystem.

Game engine optimization — access to the latest features and reduced time-to-market: QTI works closely with the world's leading third-party game engine providers, such as Unity Technologies and Epic Games, to optimize their engines for Snapdragon processors. Additionally, QTI helps expose the latest graphics APIs and advanced features of Adreno GPUs to developers through these engines. For example, QTI worked closely with Epic Games to optimize Unreal Engine's advanced lighting and post processing pipeline for Snapdragon processors.



"Bringing Unreal Engine 4 PC AAA graphics to mobile has enabled us to do content that we've never been able to do before. A big advantage has been our close relationship with Qualcomm [Technologies]. Without that close relationship we wouldn't have been able to reach this point."

Niklas "Smedis" Smedberg Senior Engine Programmer – Mobile Graphics Epic Games

Figure 13: Unreal Engine 4 demo showcasing optimized post-processing on Adreno GPU

QTI also worked with Unity Technologies to help accelerate the support of new features, making the Unity Engine one of the first gaming engines to support important OpenGL ES 3.0 features like ETC2 textures, multiple render targets (MRT), and transform feedback in Unity 4.x. Unity and QTI continue this work by helping bring new Unity 5.0 innovative features like physically based shading to Snapdragon processors.



"We're excited to work with an innovative global mobile technology provider such as Qualcomm Technologies and support their incredible Snapdragon processors, which are at the heart of many Android and Windows mobile devices across the globe."

David Hegalson CEO Unity Technologies

Figure 14: Unity 5 demo showcasing physically based shading technique on Adreno GPU

By using these popular game engines, which have been optimized for Snapdragon processors, developers can focus their time on content creation.

In addition, QTI also works closely with developers to help optimize their proprietary in-house game engines so that they run well on Snapdragon processors. In turn, these developers can then utilize their optimized engines across their internal studios when developing mobile games.

Game development tools — developing, debugging, and optimizing games: For those who are programming directly to the hardware and not using a third-party game engine, QTI has created (and made available) powerful game development tools, which are designed to help debug and optimize games, while reducing time-to-market. These tools include the <u>Adreno SDK</u>, <u>Adreno Profiler</u>, and <u>Trepn™ Profiler</u>.

The Adreno SDK includes tools, emulators, libraries, documentation, samples, and tutorials. The desktop OpenGL ES emulator is designed to eliminate the need to have a device early in the development process. Adreno SDK supports the most common APIs such as OpenGL ES, DirectX, and OpenCL. In addition, the Adreno SDK contains several time-saving utilities and over 100 samples and tutorials, including 50 advanced shader effects.

The Adreno Profiler provides developers with detailed GPU utilization analysis to help them optimize their games for faster frame rates, smoother rendering, and longer battery life. It works on commercial devices without making changes within a game, device drivers, or builds, which helps further save development time and reduce setup complexity.



Figure 15: Adreno Profiler helps optimize games

The Trepn Profiler is designed to integrate directly into a developer's workflow, which helps him or her see how much power a game is consuming, pinpoint issues and quickly resolve them. Consumers expect all day battery life, consistent performance, and a device that stays cool. In fact, battery life has become an extremely important decision factor for consumers when buying smartphones. Managing power consumption and remaining within the thermal envelope of mobile devices are key development considerations. Exceeding the thermal envelope will not only heat up the device, but also throttle the game's frame rate, which negatively affects gameplay.

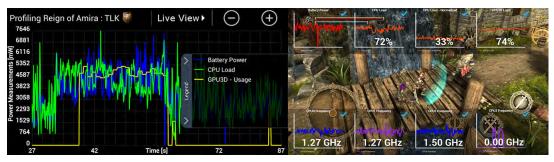


Figure 16: Trepn Profiler provides diagnostic views, such as battery power, CPU load, and GPU usage



"I would definitely recommend developers work with Qualcomm [Technologies] and use the Snapdragon tools. I know they've been a big go-to for me."

Eric Froemling

Figure 17: BombSquad by Eric Froemling

Development devices — optimizing games for real hardware:

Mobile development platforms (MDP) powered by certain Snapdragon processors in the form of tablets and smartphones are currently available to game developers prior to the launch of commercial OEM devices. These MDPs provide early access to new features of Snapdragon processors and are designed to allow developers to optimize their games in advance of the commercial launch of devices with Snapdragon processors. Devices with Snapdragon processors are available across tiers, and developers can select from a wide range of existing commercial devices to complete final testing and validation of their games.



Figure 18: Snapdragon Mobile Development Platform



"Developing on the kits is a great help for us as a development team because it allows us to address a wide installed base since many devices use the same Snapdragon chipset."

Francois Bodson Studio Manager Ubisoft Paris / Mobile

Figure 19: Assassin's Creed Pirates by Ubisoft Entertainment

5.2 Supporting the developer community

QTI can provide game developers with technical support to shorten time-to-market and improve game quality. QTI also can provide business and marketing support to help developers promote their games and facilitate potential business opportunities.

Technical support for developers — **providing expert game optimization:** Such technical support is provided via QTI's internal game studio, which is comprised of developers with comprehensive prior work experience in console, PC, and mobile gaming. Support activities range from tools, training, technical feedback on optimizations and builds (from alpha to beta to release candidates), and new feature support.

To take advantage of the capabilities of Snapdragon processors and to extend, enhance, and differentiate their games, QTI can help developers add full screen post-processing effects, OpenGL ES 3.0/3.1 features, and more. There is also a <u>developer forum</u> where developers can ask questions and engage with the QTI team and others in the Snapdragon developer community.



"The Qualcomm® Snapdragon™ processor has allowed us to create a better experience in our game Gates of Osiris because it's allowed us to develop faster, push our graphics, and understand how to optimize our game to be the best it can possibly be."

Brian McRae CEO Fenix Fire

Figure 20: QTI worked closely with Fenix Fire to optimize Gates of Osiris on Snapdragon

Business and marketing support for developers — facilitating business opportunities: QTI is engaged with some of the world's leading publishers and developers, from traditional console and PC to indies. The list includes companies such as EA, Gameloft, Activision Blizzard, Ubisoft, Square Enix, Mojang, and Take-Two Interactive. QTI also facilitates introductions for engagements with OEMs and carriers across the global gaming ecosystem.

Throughout the year, QTI has a presence at several key mobile and gaming tradeshows, where it helps developers obtain exposure to show attendees, press, and OEMs. In the past year, QTI participated in shows such as CES, MWC, GDC, E3, SIGGRAPH, Unite, and many more, reaching hundreds of thousands of attendees. In addition, QTI often features games on the latest Snapdragon development platforms and commercial devices with Snapdragon processors.





Figure 21: Showcasing games at the QTI booth during CES 2014 (left); GDC 2013 (right)

5.3 Next-generation gaming experiences

QTI continues to push the mobile gaming industry forward by developing new and innovative technologies.

QTI actively develops tech demos to showcase the latest Snapdragon features and advanced capabilities. The "Swimmer" demo is a recent example of how developers can utilize OpenGL ES 3.0 APIs for advanced rendering techniques. The demo shows advanced skin rendering using subsurface scattering techniques running optimally at 60 frames per second, at 2K resolutions.



Figure 22: QTI "Swimmer" technology demo

To help further advance the evolution of gaming, QTI and its affiliates have developed (and continue to develop) innovative technologies that support ever-more advanced capabilities and features including vision-based augmented reality (Qualcomm® Vuforia™), proximal peer-to-peer networking (AllJoyn™), biometrics, and more. For example, Vuforia supports the Smart Terrain™ feature, which is designed to provide real-time 3D mapping of a physical play area, including intelligent interaction with objects and surfaces. With this technology, gamers can create user-generated, playable content from physical objects by using their mobile device as a level editor. Another example of a promising new technology that can be utilized by game designers is biometrics data from a wearable. By capturing measurements from pulse and blood pressure sensors, gameplay can be adjusted accordingly to either reward or penalize a player.

6 Conclusion

Consumer demand for gaming on mobile devices is growing rapidly, generating strong momentum and opportunity. The Android platform is thriving and offers game developers a growing opportunity to generate revenue on a global scale. QTI is helping the mobile gaming industry to take advantage of this opportunity by solving three key challenges for developers:

- To support immersive gaming experiences within the power and thermal constraints of mobile devices, QTI creates low-power SoCs by taking a heterogeneous computing approach and designing an efficient SoC architecture.
- To efficiently address a sizeable portion of the Android gaming segment with minimal code variations, QTI's scalable architecture is designed to provide a consistent development platform across tiers.
- To take advantage of Snapdragon processors' capabilities and to help reduce development time and costs, QTI offers developers a comprehensive set of tools, support, and advanced technologies.

Snapdragon processors are purpose-built and custom designed with mobile in mind, supporting gaming experiences from simple casual games to console-quality games, so consumers can play longer and recharge less. This is yet another example of how QTI is once again re-inventing the mobile world we live in.

References

Developer tools and technologies:

- Adreno SDK: https://developer.qualcomm.com/mobile-development/maximize-hardware/mobilegaming-graphics-adreno/tools-and-resources
- 2. <u>Adreno Profiler</u>: https://developer.qualcomm.com/mobile-development/maximize-hardware/mobile-gaming-graphics-adreno/tools-and-resources
- 3. <u>Trepn Profiler</u>: https://developer.qualcomm.com/mobile-development/increase-app-performance/trepn-profiler
- 4. <u>Mobile Development Platforms</u>: https://developer.qualcomm.com/mobile-development/development-devices/snapdragon-mobile-development-platform-mdp
- Snapdragon SDK: https://developer.qualcomm.com/mobile-development/add-advanced-features/snapdragon-sdk-android
- 6. <u>Snapdragon LLVM</u>: https://developer.qualcomm.com/mobile-development/increase-app-performance/snapdragon-llvm-compiler-android
- 7. <u>Snapdragon Performance Visualizer</u>: https://developer.qualcomm.com/mobile-development/increase-app-performance/snapdragon-performance-visualizer

Developer support:

- 1. Questions/feedback/contact QTI: https://developer.qualcomm.com/contact. Use the "Mobile Gaming & Graphics Optimization (Adreno)" recipient option.
- 2. <u>Adreno developer forum</u>: https://developer.qualcomm.com/forums/qdevnet-forums/mobile-gaming-graphics-adreno
- 3. <u>Developer guide</u>: https://developer.qualcomm.com/download/adreno-sdk.zip

SoC technologies:

1. FlexRender: http://www.qualcomm.com/media/videos/flexrender-rendered-useful

Advanced technologies:

- Vuforia: https://developer.qualcomm.com/mobile-development/add-advanced-features/augmented-reality-vuforia
- 2. <u>AllJoyn</u>: https://developer.qualcomm.com/mobile-development/create-connected-experiences/peer-peer-alljoyn

Snapdragon devices:

- 1. <u>Smartphones</u>: http://www.qualcomm.com/snapdragon/smartphones
- 2. Tablets: http://www.qualcomm.com/snapdragon/tablets-pcs

1	`include "header.v"	1	SQ_SP_interp_gen_i0, SQ_SP_interp_valid, SQ_SX_interp_flat_vtx,
2	//	2	SQ_SX_interp_flat_gouraud, SQ_SX_interp_cyl_wrap, SQ_SX_pc_ptr0,
3	//	3	SQ_SX_pc_ptr1, SQ_SX_pc_ptr2, SQ_SX_rt_sel, SQ_VGT_rtr,
4	// \$Id: //depot/r400/devel/parts_lib/src/gfx/sq/sq.v#62 \$	4	SQ_SP_vsr_data, SQ_SP_vsr_double, u0_SQ_SP_vsr_valid,
5	//	5	u1_SQ_SP_vsr_valid, u2_SQ_SP_vsr_valid, u3_SQ_SP_vsr_valid,
6	// \$Change: 44294 \$	6	SQ_SP_vsr_vu_valid, SQ_SP_vsr_read, SQ_CP_vs_event,
7	II	7	SQ_CP_vs_eventid, SQ_CP_ps_event, SQ_CP_ps_eventid, SQ_TP_send,
8	// Copyright: Trade secret of ATI Technologies, Inc.	8	$SQ_TP_instr, SQ_TP_const, SQ_TP_gpr_phase, SQ_TP_gpr_wr_addr,$
9	(Copyright 2001-2002, ATI Technologies, Inc., (unpublished)	9	$SQ_TP_thread_id, u0_SQ_TP_lod_correct, u0_SQ_TP_pix_mask,$
10	II	10	u1_SQ_TP_lod_correct, u1_SQ_TP_pix_mask, u2_SQ_TP_lod_correct,
11	// All rights reserved. This notice is intended as a precaution against	11	u2_SQ_TP_pix_mask, u3_SQ_TP_lod_correct, u3_SQ_TP_pix_mask,
12	// inadvertent publication and does not imply publication or any waiver	12	$SQ_SP_gpr_wr_addr, u0_SQ_SP_gpr_wr_en, u1_SQ_SP_gpr_wr_en,$
13	// of confidentiality. The year included in the foregoing notice is the	13	u2_SQ_SP_gpr_wr_en, u3_SQ_SP_gpr_wr_en, SQ_SP_gpr_rd_addr,
14	// year of creation of the work.	14	SQ_SP_gpr_rd_en, SQ_SP_gpr_phase_mux, SQ_SP_channel_mask,
15	//	15	u0_SQ_SP_pix_mask, u1_SQ_SP_pix_mask, u2_SQ_SP_pix_mask,
16	//	16	u3_SQ_SP_pix_mask, SQ_SP_gpr_input_mux, SQ_SP_auto_count,
17	`include "register_addr.v"	17	SQ_SP_instruct_start, SQ_SP_instruct, SQ_SP_const, SQ_SP_stall,
18	`include "sq_reg.v"	18	SQ_SP_exporting, SQ_SP_exp_id, u0_SQ_SP_exp_pvalid,
	`include "sq_defs.v"	19	u1_SQ_SP_exp_pvalid, u2_SQ_SP_exp_pvalid, u3_SQ_SP_exp_pvalid,
	`include "/sq_reg_old.v"	20	SQ_SX_pc_wr_addr, SQ_SX_pc_wr_en, SQ_SX_pc_channel_mask,
21		21	SQ_SX_exp_type, SQ_SX_exp_number, SQ_SX_exp_id, SQ_SX_exp_valid,
	module sq (/*AUTOARG*/	22	SQ_SX_exp_state, SQ_SX_free_done, SQ_SX_free_id, SQ_RBB_rd,
23	// Outputs	23	SQ_RBB_rs, SQ_RBBM_nrtrtr, SQ_RBBM_rtr, SQ_RBBM_cntx0_busy,
24	SQ_SC_free_buff, SQ_SC_dec_cntr_cnt, SQ_SP_interp_prim_type,	24	SQ_RBBM_cntx17_busy, SQ_CG_threshold_hi, SQ_CG_threshold_lo,
25	SQ_SP_interp_mode, SQ_SP_interp_ijline, SQ_SP_interp_buff_swap,	25	// Inputs
	«Co. ZkZ		
	Page 1 of 92 Ex. 2093 - sq.v		Page 2 of 92 Ex. 2093 - sq.v
	20. 2070 04.1		200 2000 04.1
2 3 4 5 6 7 8 9 10	VGT_SQ_vsisr_continued, VGT_SQ_event, VGT_SQ_end_of_vtx_vect, VGT_SQ_indx_valid, VGT_SQ_state, VGT_SQ_send, TP_SQ_type, TP_SQ_data_rdy, TP_SQ_thread_id, TP_SQ_fetch_stall, u0_SX_SQ_exp_count_rdy, u0_SX_SQ_exp_pos_avail, u0_SX_SQ_exp_buf_avail, u1_SX_SQ_exp_count_rdy, u1_SX_SQ_exp_pos_avail, u1_SX_SQ_exp_buf_avail, u0_SP_SQ_const_addr, u0_SP_SQ_valid, u1_SP_SQ_const_addr, u1_SP_SQ_valid, u2_SP_SQ_const_addr, u2_SP_SQ_valid, u3_SP_SQ_const_addr, u3_SP_SQ_valid, u0_SP_SQ_kill_vect, u1_SP_SQ_kill_vect, u2_SP_SQ_valid, u2_SP_SQ_kill_vect,	2 3 4 5 6 7 8 9 10	input [3:0] SC_SQ_quad_mask; input [0:0] SC_SQ_end_of_prim; input [2:0] SC_SQ_estate_id; input [15:0] SC_SQ_pix_mask; input [2:0] SC_SQ_prim_type; input [10:0] SC_SQ_pe_ptr0; // 2nd cycle fields input [10:0] SC_SQ_pe_ptr1; input [10:0] SC_SQ_pe_ptr2; input [23:0] SC_SQ_pe_ptr2; input [23:0] SC_SQ_lod_correct;
12	RBBM_a, RBBM_wd, RBBM_we, RBBM_re, RBB_rs, RBB_rd, RBBM_be,	12	*/
13	RBBM_SQ_soft_reset, CG_SQ_pm_enb, sclk_global, srst	13	
14);	14	output [0:0] SQ_SC_free_buff;
15	//	15	output [0:0] SQ_SC_dec_cntr_cnt;
16		16	
17	// SC-SQ IJ control interface	17	u,
18	//	18	//
19	input [53:0] SC_SQ_data;	19	// SQ-SP/SX Interpolator Bus interface
20	input [0:0] SC_SQ_valid;	20	//
21	/*	21	output [2:0] SQ_SP_interp_prim_type;
22	// 1st cycle fields	22	output [0:0] SQ_SP_interp_mode;
23	input [0:0] SC_SQ_event;	23	output [1:0] SQ_SP_interp_ijline;
24	input [3:0] SC_SQ_event_id;	24	output [0:0] SQ_SP_interp_buff_swap;
25	input [2:0] SC_SQ_pc_dealloc;	25	output [0:0] SQ_SP_interp_gen_i0;
	Page 3 of 92		Page 4 of 92
	Ex. 2093 - sq.v		Ex. 2093 - sq.v

ATI 2093 LG v. ATI IPR2015-00326

1	output [0:0] SQ_SP_interp_valid;		1	input [0:0] VGT_SQ_send;	
2			2		
3	output [1:0] SQ_SX_interp_flat_vtx;		3	output [0:0] SQ_VGT_rtr;	
4	output [0:0] SQ_SX_interp_flat_gouraud;		4		
5	output [3:0] SQ_SX_interp_cyl_wrap;		5		
6			6	//	
7			7	// SQ-SP Vertex Interface	
8	//		8	//	
9	// SQ-SX Parameter Cache Read control		9	output [95:0] SQ_SP_vsr_data; output [0:0] SQ_SP_vsr_double;	
11	output [10:0] SQ_SX_pc_ptr0;		11	output [0:0] u0_SQ_SP_vsr_valid;	
12	output [10:0] SQ_SX_pc_ptr1;		12	output [0:0] u1_SQ_SP_vsr_valid;	
13	output [10:0] SQ_SX_pc_ptr2;		13	output [0:0] u2_SQ_SP_vsr_valid;	
14	output [0:0] SQ_SX_rt_sel;		14	output [0:0] u3_SQ_SP_vsr_valid;	
15			15	output [3:0] SQ_SP_vsr_vu_valid;	
16			16	output [0:0] SQ_SP_vsr_read;	
17	//		17		
18	// VGT-SQ Vertex Interface		18		
19	//		19	//	
20	input [95:0] VGT_SQ_vsisr_data;		20	// SQ-CP Event Status	
21	input [0:0] VGT_SQ_vsisr_continued;		21	//	
22	input [0:0] VGT_SQ_event;		22	output [0:0] SQ_CP_vs_event;	
23	input [0:0] VGT_SQ_end_of_vtx_vect;		23	output [1:0] SQ_CP_vs_eventid;	
24	input [0:0] VGT_SQ_indx_valid;		24	output [0:0] SQ_CP_ps_event;	
25	input [2:0] VGT_SQ_state;		25	output [1:0] SQ_CP_ps_eventid;	
	Page 5 of 92	Ex. 2093 - sq.v		Page 6 of 92	Ex. 2093 - sq.v
1			1	output [5:0] u3_SQ_TP_lod_correct;	
1 2			1 2	output [5:0] u3_SQ_TP_lod_correct; output [3:0] u3_SQ_TP_pix_mask;	
	//				
2	//		2		
2			2 3	output [3:0] u3_SQ_TP_pix_mask;	
2 3 4	// SQ-TP (Texture Pipe) (part of this interface is broadcast)		2 3 4	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type;	
2 3 4 5 6 7	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id;	
2 3 4 5 6 7 8	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy;	
2 3 4 5 6 7 8	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id;	
2 3 4 5 6 7 8 9	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall;	
2 3 4 5 6 7 8 9 10	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall;	
2 3 4 5 6 7 8 9 10 11	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall;	
2 3 4 5 6 7 8 9 10 11 12	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall;	
2 3 4 5 6 7 8 9 10 11 12 13 14	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	// SQ-TP (Texture Pipe) (part of this interface is broadcast) //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	output [3:0] u3_SQ_TP_pix_mask; input [0:0] TP_SQ_type; input [0:0] TP_SQ_data_rdy; input [5:0] TP_SQ_thread_id; input [0:0] TP_SQ_fetch_stall; //	



```
wire [('SQ_PROGRAM_REG_COUNT_SHADING_PS_NUM_REG_SIZE * 8) - 1: 0] ps_num_reg_set;
2
      // MISC
                                                                                                                   wire [(`SQ_PROGRAM_REG_COUNT_SHADING_VS_NUM_REG_SIZE * 8) - 1: 0]
                                                                                                                   wire [('SQ_PROGRAM_REG_COUNT_SHADING_PARAM_SHADE_SIZE * 8) - 1: 0]
      input [0:0] CG_SQ_pm_enb;
                                                                                                                   wire [('SQ_INST_STORE_MANAGMENT_INST_BASE_VTX_SIZE) - 1: 0]
      output [0:0] SQ_CG_threshold_hi;
      output [0:0] SQ_CG_threshold_lo;
                                                                                                            10
11
                                                                                                                   wire [(`SQ_INST_STORE_MANAGMENT_INST_BASE_PIX_SIZE) - 1: 0]
                                                                                                                   wire [(`SQ_WRAPPING_1_REG_SIZE * 8) - 1: 0] sq_wrapping_1_set;
      input [0:0] sclk_global;
11
      input [0:0] srst;
                                                                                                                   wire [(`SQ_WRAPPING_0_REG_SIZE * 8) - 1: 0]
                                                                                                                        sq wrapping 0 set;
12
                                                                                                                   wire [(`SQ_SAMPLING_MODE_SAMPLING_MODE_SIZE * 8) - 1: 0]
13
                                                                                                             18
                                                                                                                   wire [('SO VS EXPORT COUNT COUNT7 SIZE * 8) - 1: 0]
15
                                                                                                                   wire [('SQ IMPORTS EXPORTS GEN INDEX SIZE * 8) - 1: 0]
17
                                                                                                            21
18
      // -- Module Instantiations --
                                                                                                            22
23
                                                                                                                   wire [(`SQ_IMPORTS_EXPORTS_PARAM_GEN_I0_SIZE * 8) - 1: 0]
                                                                                                                        param_gen_i0_set;
19
                                                                                                            24
25
                                                                                                                   wire [('SQ_PROGRAM_CNTL_VS_RESOURCE_SIZE * 8) -1: 0]
20
21
22
     "
                                                                                                            26
                                                                                                                   wire [CSO PROGRAM CNTL PS RESOURCE SIZE * 8) - 1: 0] ps resource set:
23
                                                                                                            27
24
      parameter HI = 1'b1;
                                                                                                                   wire [(`SQ_PROGRAM_CNTL_VS_EXPORT_COUNT_SIZE * 8) - 1: 0]
25
      parameter LO = 1'b0;
                                                                                                                  wire [(`SQ_PROGRAM_CNTL_VS_EXPORT_MODE_SIZE * 8) - 1: 0]
26
27
      // - wires for local registers
                                        Page 13 of 92
                                                                                                                                                    Page 14 of 92
                                                                        Ex. 2093 - sq.v
                                                                                                                                                                                    Ex. 2093 - sq.v
       wire [('SQ_PROGRAM_CNTL_PS_EXPORT_MODE_SIZE * 8) - 1: 0]
                                                                                                                  wire [0:0] vtx write gnt;
                                                                                                             2
                                                                                                                   wire [0:0] vtx write req;
      wire [('SQ_PS_PROGRAM_BASE_SIZE * 8) - 1: 0] ps_program_base_set;
                                                                                                                   wire [0:0] vtx_write_busy;
      wire [('SQ_PS_PROGRAM_BASE_SIZE * 8) - 1: 0] vs_program_base_set;
                                                                                                                   wire [6:0] gpr base;
                                                                                                             7
                                                                                                                   wire [00:0] vism_ctl_pkt_rts;
       // -- Vertex Input Control --
                                                                                                                   wire [11:0] vism_instr_ptr;
                                                                                                                   wire [63:0] vism_valid_bits;
10
                                                                                                            10
                                                                                                                   wire [06:0] vism_gpr_base;
11
                                                                                                                   wire [02:0] vism_context_id;
12
                                                                                                                   wire [00:0] vism_resource;
13
                                                                                                                   wire [00:0] vism_first_thread;
     //wire [8*6-1:0] vs_num_reg_set; // connected SQ PROGRAM REG COUNT SHADING.VS NUM REG (6 bits)
14
15
                                                                                                                   wire [00:0] vtb_rtr;
       //wire [8*1-1:0] gen_index_set; // connected to SQ_IMPORTS_EXPORTS.GEN_INDEX
16
17
                                                                                                                  wire [6:0] vi_gpr_wr_addr;
18
                                                                                                                   wire [0:0] vi_gpr_wr_en;
19
      wire [0:0] vtx_alloc_req;
                                                                                                                   wire [7:0] acs context valid;
20
      wire [5:0] vtx_alloc_space;
21
      wire [0:0] vtx_alloc_ack;
                                                                                                            20
22
                                                                                                            21
23
      wire [0:0] vtx_dealloc_req;
                                                                                                                  // -- Vertex Input State Machine --
                                                                                                            22
24
      wire [5:0] vtx_dealloc_space;
                                                                                                            23
25
       wire [0:0] vtx_dealloc_ack;
                                                                                                            24
26
                                                                                                            25
                                                                                                                   sq_vism
                                        Page 15 of 92
                                                                                                                                                    Page 16 of 92
                                                                        Ex. 2093 - sq.v
                                                                                                                                                                                    Ex. 2093 - sq.v
```

```
1
      u_sq_vism
                                                                                                                        .i_vs_base_set (vs_program_base_set),
2
                                                                                                                        //.i_vs_resource_set (vs_resource_set),
                                                                                                                        .i_vs_resource_set (8'b11111111),
       // VGT Interface
        .i_vgt_vsisr_data
                             (VGT_SQ_vsisr_data),
        .i\_vgt\_vsisr\_double \qquad (VGT\_SQ\_vsisr\_continued),\\
                                                                                                                        // input arbiter interface
        .i_vgt_end_of_vector (VGT_SQ_end_of_vtx_vect),
                                                                                                                        .o_v_gpr_wrt_req
                                                                                                                                           (vtx_write_req),
        .i_vgt_indx_valid (VGT_SQ_indx_valid),
                                                                                                                        .i_v_gpr_wrt_grant (vtx_write_gnt),
        .i_vgt_vsisr_state (VGT_SQ_state),
                                                                                                                        .o_vism_busy
                                                                                                                                           (vtx_write_busy),
        .i_vgt_send (VGT_SQ_send),
                                                                                                                       // gpr alloc Interface
        .o\_vgt\_rtr \hspace{1cm} (SQ\_VGT\_rtr),
11
                                                                                                                       .o_v_gpr_space_req (vtx_alloc_req),
12
        // SP Interface
                                                                                                                       .o_v_gpr_space
                                                                                                                                             (vtx_alloc_space),
13
                             (SQ_SP_vsr_data),
                                                                                                                       .i_v_gpr_space_grant (vtx_alloc_ack),
        .o sp vsr data
14
        .o_sp_vsr_double
                         (SQ_SP_vsr_double),
                                                                                                                       .i_v_gpr_base_addr (gpr_base),
     .o\_sp\_vsr\_valid \\ ul\_SQ\_SP\_vsr\_valid, u0\_SQ\_SP\_vsr\_valid,),
                                                                  u2 SQ SP vsr valid,
                                                                                                               15
                                                                                                               16
                                                                                                                       // control packet to VTB
17
        .o_sp_vsr_vu_valid (SQ_SP_vsr_vu_valid),
                                                                                                               17
                                                                                                                        .o v ld cntl pkt (vism ctl pkt rts),
18
         .o_sp_vsr_read
                             (SQ_SP_vsr_read),
                                                                                                                        .o_vs_first_thread (vism_first_thread),
                                                                                                               18
19
                                                                                                                        .o_vs_resource (vism_resource),
                                                                                                               19
20
        // to output mux (currently called ais_output), then to SP
                                                                                                               20
                                                                                                                        .o_vs_instr_ptr
                                                                                                                                        (vism instr ptr),
21
         .o_v_gpr_addr (vi_gpr_wr_addr), // VISM gpr write address
                                                                                                               21
                                                                                                                        .o vector valid
                                                                                                                                         (vism valid bits).
22
         .o_v_gpr_we
                             (vi_gpr_wr_en),
                                                    // VISM gpr write enable
                                                                                                                        .o_v_gpr_base
                                                                                                                                             (vism_gpr_base),
23
                                                                                                               23
                                                                                                                        .o_vgt_state
                                                                                                                                            (vism_context_id),
24
        // local register inputs
                                                                                                                        .i_vtb_rtr (vtb_rtr),
                                                                                                               24
25
        .i_vs_num_reg (vs_num_reg_set),
                                                                                                               25
                                                                                                                       .i_context_valid (acs_context_valid),
        .i_gen_index
                       (gen_index_set),
                                       Page 17 of 92
                                                                                                                                                        Page 18 of 92
                                                                          Ex. 2093 - sq.v
                                                                                                                                                                                         Ex. 2093 - sq.v
        .i_clk
                              (sclk_global),
2
        .i reset
                              (srst)
                                                                                                                     wire [0:0] pix alloc req;
                                                                                                                      wire [5:0] pix_alloc_space;
                                                                                                                      wire [0:0] pix alloc ack;
                                                                                                                     wire [0:0] pix_dealloc_req;
                                                                                                                      wire [5:0] pix_dealloc_space;
      // -- Pixel Input Control --
                                                                                                                      wire [0:0] pix_dealloc_ack;
                                                                                                                     wire [0:0] pix_write_gnt;
10
                                                                                                               10
11
                                                                                                               11
                                                                                                                     wire [0:0] pix_write_req;
12
      // - interconnect wires
                                                                                                               12
                                                                                                                      wire [0:0] pix_write_busy;
13
                                                                                                               13
14
      wire [0:0] pb_rts;
                                                                                                                     wire [6:0] pi_gpr_wr_addr;
      wire ['SQ_PB_WIDTH-1:0] pb_rd_data0;
                                                                                                               15
                                                                                                                      wire [3:0] pi_gpr_wr_en;
16
      wire [`SQ_PB_WIDTH-1:0] pb_rd_data1;
17
      wire ['SQ_PB_WIDTH-1:0] pb_rd_data2;
                                                                                                               17 wire [00:0] pism_ctl_pkt_rts;
18
      wire ['SQ_PB_WIDTH-1:0] pb_rd_data3;
                                                                                                               18 wire [143:0] pism_lod_correct;
19
                                                                                                               19 wire [11:0] pism_instr_ptr;
20
      wire [1:0] pb max went;
                                                                                                               20 wire [63:0] pism valid bits;
21
      wire [2:0] pb pix state;
                                                                                                               21 wire [06:0] pism gpr base;
22
                                                                                                               22 //wire [02:0] pism_context_id;
23
                                                                                                               23
      wire [0:0] pb read en;
                                                                                                                     wire [00:0] pism resource;
24
      wire [0:0] free buff;
                                                                                                               24
                                                                                                                      wire [00:0] pism_first_thread;
25
      wire [0:0] pi_rtr;
                                                                                                                      wire [00:0] ptb_rtr;
                                         Page 19 of 92
                                                                                                                                                        Page 20 of 92
                                                                          Ex. 2093 - sq.y
                                                                                                                                                                                         Ex. 2093 - sq.v
```

```
.SC SQ valid (SC SQ valid),
2
      wire [0:0] pb_rd_en;
3
                                                                                                                          .SQ SC free buff(SQ SC free buff),
4
      wire [0:0] vtx_vector_done;
                                                                                                                          .SQ_SC_dec_cntr_cnt(SQ_SC_dec_cntr_cnt),
      wire [0:0] pb_event_rts;
                                                                                                                          // inputs from PISM
      wire [3:0] pb_event_id;
                                                                                                                                      (pi_rtr),
      wire [2:0] pb_event_state;
                                                                                                                          .pi_read_en (pi_read_en),
       wire [2:0] pb_dealloc_cnt;
                                                                                                                          .pi_free_buff (pi_free_buff),
       wire [0:0] pb_dealloc_vld;
11
                                                                                                                          // outputs to PISM
12
      wire [1:0] gpr_phase;
                                                                                                                          .pb_rts
                                                                                                                                          (pb_rts),
13
      wire [1:0] is_phase;
                                                                                                                          .pb_rd_data0 (pb_rd_data0),
       wire [1:0] is_subphase;
                                                                                                                          .pb_rd_data1 (pb_rd_data1),
15
                                                                                                                  15
                                                                                                                          .pb_rd_data2 (pb_rd_data2),
16
                                                                                                                  16
                                                                                                                           .pb\_rd\_data3 \qquad (pb\_rd\_data3),
17
                                                                                                                  17
      // -- Pointer Buffer --
18
                                                                                                                  18
                                                                                                                          .pb_max_went (pb_max_went),
19
                                                                                                                  19
                                                                                                                          .pb_pix_state (pb_pix_state),
20
21
      sq_ptr_buff
                                                                                                                  21
                                                                                                                           .pb_event_rts (pb_event_rts),
22
       u_sq_ptr_buff
                                                                                                                          .pb_event_id
                                                                                                                                         (pb_event_id),
23
                                                                                                                  23
                                                                                                                          .pb_event_state (pb_event_state),
24
       // SC interface
        .SC_SQ_data (SC_SQ_data),
                                                                                                                          .pb_dealloc_cnt (pb_dealloc_cnt),
                                         Page 21 of 92
                                                                                                                                                           Page 22 of 92
                                                                            Ex. 2093 - sq.v
                                                                                                                                                                                             Ex. 2093 - sq.v
         .pb_dealloc_vld (pb_dealloc_vld),
                                                                                                                          .pb pix state (pb pix state),
 2
        .vtx vector done (vtx vector done),
                                                                                                                           .pb event rts (pb event rts),
                                                                                                                           .pb event id (pb event id),
         .pix_write_busy (pix_write_busy),
                                                                                                                           .pb_event_state (pb_event_state),
         .clk(sclk global),
 7
         .reset(srst)
                                                                                                                           .pi read en
                                                                                                                                          (pi_read_en),
 8
                                                                                                                           .pi_free_buff (pi_free_buff),
                                                                                                                                          (pi_rtr),
10
                                                                                                                  10
11
                                                                                                                  11
                                                                                                                          // interfaces to gpr alloc and input arb
12
      // -- Pixel Input State Machine --
                                                                                                                          .pix_alloc_req (pix_alloc_req),
13
                                                                                                                          .pix_alloc_space (pix_alloc_space),
14
                                                                                                                          .pix_alloc_ack (pix_alloc_ack),
15
                                                                                                                          .pix_alloc_base (gpr_base),
       sq_pism
16
       u_sq_pism
17
                                                                                                                          .pix_write_req (pix_write_req),
18
       // pointer buffer interface
                                                                                                                          .pix write busy (pix write busy),
19
                                                                                                                          .pix_write_gnt (pix_write_gnt),
       .pb rts
       .pb_rd_data0 (pb_rd_data0),
20
                                                                                                                  21
21
      .pb rd datal (pb rd datal),
                                                                                                                          // inputs from local registers
22
      .pb_rd_data2 (pb_rd_data2),
                                                                                                                  22 .num_reg_set (ps_num_reg_set), //
23 SQ_PROGRAM_CNTL.PS_NUM_REG (6 bits)
                                                                                                                                                                               connected
                                                                                                                                                                                                         to
23
        .pb_rd_data3 (pb_rd_data3),
                                                                                                                      .num_param_set (vs_export_count_set), //
SQ_PROGRAM_CNTL.VS_EXPORT_COUNT (4 bits)
                                                                                                                                                                                   connected
24
25
        .pb_max_went (pb_max_went),
                                         Page 23 of 92
                                                                                                                                                            Page 24 of 92
                                                                            Ex. 2093 - sq.v
                                                                                                                                                                                             Ex. 2093 - sq.v
```

```
.param_shade_set (param_shade_set), //
SQ_INTRPOLATOR_CNTL.PARAM_SHADE (16 bits)
                                                                         connected
                                                                                                                                                 .pix_gpr_wr_en (pi_gpr_wr_en),
      .param_gen_i0_set (param_gen_i0_set), //
SQ_PROGRAM_CNTL.PARAM_GEN_I0 (1 bit)
                                                                             connected
                                                                                                                                                 // outputs to SP interp ctl
      //.sampling_mode_set(sampling_mode_set), //
SQ_SAMPLING_MODE.SAMPLING_MODE (1 bit)
                                                                                                                                                 .SQ_SP_interp_prim_type
                                                                                                                                                                               (SQ_SP_interp_prim_type),
                                                                           connected
                                                                                                                                                 . SQ\_SP\_interp\_mode \qquad (SQ\_SP\_interp\_mode), \ \ //\ ????
      .sq\_wrapping\_0\_set(sq\_wrapping\_0\_set), \hspace{1cm} /\!/ \hspace{1cm} connected \hspace{1cm} to \hspace{1cm} SQ\_WRAPPING\_0 \hspace{1cm} (32 \hspace{1cm} bits)
                                                                                                                                                 . SQ\_SP\_interp\_ijline \qquad (SQ\_SP\_interp\_ijline),
      .sq\_wrapping\_1\_set(sq\_wrapping\_1\_set), \hspace{1cm} /\!\!/ \hspace{1cm} connected \hspace{1cm} to \hspace{1cm} SQ\_WRAPPING\_1 \hspace{1cm} (32 \hspace{1cm} bits)
                                                                                                                                                  . SQ\_SP\_interp\_buff\_swap \qquad \qquad (SQ\_SP\_interp\_buff\_swap),
                                                                                                                                                 .SQ_SP_interp_gen_i0 (SQ_SP_interp_gen_i0),
11
                                                                                                                                                 .SQ_SP_interp_valid (SQ_SP_interp_valid),
      \label{lem:connected} $$p_program\_base\_set(ps\_program\_base\_set) // $$ connected to $$Q_PS\_PROGRAM.BASE (12 bits)$
12
13
                                                                                                                                                 . SQ\_SX\_interp\_flat\_vtx \quad (SQ\_SX\_interp\_flat\_vtx),
      .ps_resource_set (ps_resource_set), //
SQ_PROGRAM_CNTL.PS_RESOURCE (1 bit)
                                                                                                                                                 . SQ\_SX\_interp\_flat\_gouraud (SQ\_SX\_interp\_flat\_gouraud),
16
                                                                                                                                       13
                                                                                                                                                 .SQ SX interp cyl wrap (SQ SX interp cyl wrap),
17
         // pix control packet outputs to pixel thread buffer
18
          .ctl pkt rts q (pism ctl pkt rts),
                                                                                                                                       15
                                                                                                                                                 // outputs to SX param cache
19
          .lod correct q (pism lod correct), // state
                                                                                                                                       16
                                                                                                                                                 .SQ_SX_pc_ptr0 (SQ_SX_pc_ptr0),
20
          .valid_bits_q (pism_valid_bits), // state
                                                                                                                                       17
                                                                                                                                                 .SQ_SX_pc_ptrl (SQ_SX_pc_ptrl),
21
          .ps_program_base_q (pism_instr_ptr), // state
                                                                                                                                       18
                                                                                                                                                 .SQ_SX_pc_ptr2 (SQ_SX_pc_ptr2),
22
          .gpr\_base\_q \qquad \quad (pism\_gpr\_base), \qquad /\!/ \ state
                                                                                                                                                 .SQ_SX_rt_sel (SQ_SX_rt_sel),
                                                                                                                                       19
23
          .ps\_resource\_q \qquad (pism\_resource), \qquad \textit{// status: resource bit : } tex=1, alu=0
                                                                                                                                       20
24
          .first_thread_q \quad (pism_first_thread), \quad /\!/ status: first thread of a new state
                                                                                                                                                 .clk(sclk_global),
                                                                                                                                       21
    .ptb_rtr
CFS update
                            (ptb_rtr), // PTB is ready when not full OR when not doing a
                                                                                                                                       22
                                                                                                                                                 .reset(srst)
                                                                                                                                       23
27
                                                                                                                                       24
28
         // outputs to ais_output
                                                                                                                                       25
          .pix_gpr_wr_addr (pi_gpr_wr_addr),
                                                Page 25 of 92
                                                                                                                                                                                         Page 26 of 92
                                                                                          Ex. 2093 - sq.v
                                                                                                                                                                                                                                 Ex. 2093 - sq.v
                                                                                                                                                 .pix_alloc_ack (pix_alloc_ack),
                                                                                                                                                 .pix_dealloc_ack (pix_dealloc_ack),
        // -- GPR Allocation and Input Arbitration --
                                                                                                                                                  .vtx alloc ack (vtx alloc ack),
                                                                                                                                                  .vtx dealloc ack (vtx dealloc ack),
                                                                                                                                                 .clk(sclk global),
                                                                                                                                        7
                                                                                                                                                  .reset(srst)
       // -- GPR Allocation --
10
                                                                                                                                       10
        sq_gpr_alloc
11
                                                                                                                                       11
12
        u_sq_gpr_alloc
                                                                                                                                       12
                                                                                                                                              // -- Input Arbitration --
13
                                                                                                                                       13
14
          .vtx_alloc_req (vtx_alloc_req),
          .vtx_alloc_space (vtx_alloc_space),
                                                                                                                                               sq_input_arb
16
          .vtx_dealloc_req (vtx_dealloc_req),
                                                                                                                                               u_sq_input_arb
17
          .vtx_dealloc_space(vtx_dealloc_space),
                                                                                                                                       17
18
                                                                                                                                       18
                                                                                                                                                 .vtx req (vtx write req),
19
          .pix alloc req (pix alloc req),
                                                                                                                                                 .vtx busy (vtx write busy),
20
          .pix_alloc_space (pix_alloc_space),
                                                                                                                                       20
                                                                                                                                                 .pix req (pix write req),
21
                                                                                                                                       21
          .pix_dealloc_req (pix_dealloc_req),
                                                                                                                                                 .pix busy (pix write busy),
22
          .pix_dealloc_space(pix_dealloc_space),
23
                                                                                                                                       23
                                                                                                                                                 .gpr_phase(gpr_phase),
24
                                                                                                                                       24
          .base_ptr
                            (gpr_base),
25
                                                                                                                                       25
                                                                                                                                                 .vtx_gnt (vtx_write_gnt),
                                                 Page 27 of 92
                                                                                                                                                                                         Page 28 of 92
                                                                                         Ex. 2093 - sq.v
                                                                                                                                                                                                                                 Ex. 2093 - sq.v
```

```
.vtx_sel (ia_vertex_sel),
when l (select pixel if 0)
                                                                                                                 wire [11:0] tif is read addr;
                                   // select VISM gpr write address, enable to drive to SP
                                                                                                                 wire [11:0] aif0_is_read_addr;
        .pix_gnt (pix_write_gnt),
                                                                                                                  wire [11:0] aif1_is_read_addr;
        .clk(sclk_global),
                                                                                                                  wire [95:0] is_read_data;
        .reset(srst)
                                                                                                                                        tcfs_update;
                                                                                                                                        tcfs_thread_id;
10
                                                                                                                  wire [`SQ_CFS_STATE_WIDTH-1:0]tcfs_state;
11
                                                                                                                  wire ['SQ_STATUS_WIDTH-1:0] tcfs_status;
       // -- Thread Buffers, Thread Arbiters, Control Flow Sequencers --
13
                                                                                                            13
14
                                                                                                                  wire [0:0]
                                                                                                                                        acfs0_update;
15
                                                                                                            15
                                                                                                                 wire [5:0]
                                                                                                                                        acfs0 thread id;
16
      // - interconnect wires
                                                                                                                  wire [`SQ_CFS_STATE_WIDTH-1:0]acfs0_state;
                                                                                                            16
17
                                                                                                            17
                                                                                                                  wire [`SQ_STATUS_WIDTH-1:0] acfs0_status;
      parameter TB DEPTH = 16:
18
                                                                                                            18
      parameter TB_ADDR_WIDTH = 4;
19
                                                                                                            19
                                                                                                                  wire [0:0]
                                                                                                                                        acfs1 update;
20
                                                                                                            20
                                                                                                                  wire [5:0]
                                                                                                                                        acfs1 thread id;
21
      wire [0:0] state_read_phase;
                                                                                                                  wire ['SQ_CFS_STATE_WIDTH-1:0]acfs1_state;
                                                                                                            21
22
      wire [1:0] cfs_phase;
                                                                                                                  wire ['SQ_STATUS_WIDTH-1:0] acfs1_status;
23
                                                                                                            23
24
      wire [11:0] tcfs_is_read_addr;
                                                                                                            24
                                                                                                                  wire [TB_DEPTH-1:0]
25
      wire [11:0] acfs0_is_read_addr;
                                                                                                                  wire [`SQ_VTX_STATE_WIDTH-1:0] vtx_tex_state;
       wire [11:0] acfs1_is_read_addr;
                                       Page 29 of 92
                                                                                                                                                   Page 30 of 92
                                                                        Ex. 2093 - sq.v
                                                                                                                                                                                    Ex. 2093 - sq.v
      wire ['SQ_STATUS_WIDTH-1:0]
                                       vtx_tex_status;
      wire [0:0] vtx_tex_winner_ack;
2
                                                                                                             2
                                                                                                                 wire [0:0]
                                                                                                                                         ais0 done;
       wire [0:0]
                                                                                                                  wire [0:0]
                                                                                                                                          ais0 thread type;
                            vtx_tex_state_vld;
       wire [TB_ADDR_WIDTH-1:0] vtx_tex_winner_q;
                                                                                                                   wire [5:0]
                                                                                                                                          ais0 thread id;
      wire [TB DEPTH-1:0]
                                                                                                                  wire [0:0]
                                                                                                                                         ais1 done;
                              pix_tex_req_q;
      wire [`SQ_PIX_STATE_WIDTH-1:0] pix_tex_state;
                                                                                                                   wire [0:0]
                                                                                                                                         ais1 thread type;
      wire [`SQ_STATUS_WIDTH-1:0] pix_tex_status;
                                                                                                                   wire [5:0]
                                                                                                                                          ais1 thread id;
       wire [0:0] pix_tex_winner_ack;
10
       wire [0:0]
                          pix_tex_state_vld;
                                                                                                            10
                                                                                                                  wire
                                                                                                                                   vtx_state_change;
11
       wire \ [TB\_ADDR\_WIDTH-1:0] \quad \  pix\_tex\_winner\_q;
                                                                                                            11
                                                                                                                  wire [2:0]
                                                                                                                                   vtx_old_state;
12
                                                                                                            12
13
      wire [TB_DEPTH-1:0] vtx_alu_req_q;
                                                                                                            13 wire
                                                                                                                                   pix_state_change;
14
       wire [`SQ_VTX_STATE_WIDTH-1:0] vtx_alu_state;
                                                                                                                  wire [2:0]
                                                                                                                                   pix_old_state;
15
       wire [`SQ_STATUS_WIDTH-1:0] vtx_alu_status;
                                                                                                            15
16
                        vtx_alu_winner_ack;
                                                                                                                 wire [0:0]
                                                                                                                                         tarb_rts;
17
       wire [0:0] vtx_alu_state_vld;
                                                                                                                 wire ['SQ_STATE_WIDTH-1:0] tarb_state;
18
      wire [TB_ADDR_WIDTH-1:0] vtx_alu_winner_q;
                                                                                                            18
                                                                                                                  wire ['SQ_STATUS_WIDTH-1:0] tarb_status;
19
                                                                                                                  wire [0:0]
                                                                                                                                      tarb_thread_type;
20
      wire [TB DEPTH-1:0]
                                                                                                            20
                               pix alu req q;
                                                                                                                  wire [0:0]
                                                                                                                                         tcfs rtr;
      wire ['SQ_PIX_STATE_WIDTH-1:0] pix_alu_state;
21
                                                                                                            21
      wire [`SQ_STATUS_WIDTH-1:0] pix_alu_status;
22
                                                                                                            22 wire [0:0]
                                                                                                                                        tcfs rts:
23
                                                                                                            23 wire ['SQ_CTL_PKT_WIDTH-1:0] tcfs_ctl_pkt;
      wire [0:0]
                            pix_alu_winner_ack;
24
                                                                                                            24 wire [11:0]
                                                                                                                                       tcfs_tgt_instr_ptr;
      wire [0:0]
                            pix_alu_state_vld;
      wire [TB_ADDR_WIDTH-1:0] pix_alu_winner_q;
                                                                                                            25
                                                                                                                 wire [11:0]
                                                                                                                                         tcfs_tgt_instr_cnt;
                                       Page 31 of 92
                                                                                                                                                   Page 32 of 92
                                                                       Ex. 2093 - sq.v
                                                                                                                                                                                    Ex. 2093 - sq.v
```

```
wire [0:0]
                                                                                                                              wire [0:0]
                                 tcfs thread type:
                                                                                                                                                        vtx tb busy;
 2
       wire [0:0]
                                 tif rtr;
                                                                                                                              wire [0:0]
                                                                                                                                                        pix tb busy;
 3
                                                                                                                               wire [0:0]
                                                                                                                                                        tcfs busy;
       wire [0:0]
                                 aarb rts0;
                                                                                                                               wire [0:0]
                                                                                                                                                        acfs0_busy;
       wire [0:0]
                                 aarb_rts1;
                                                                                                                               wire [0:0]
                                                                                                                                                        acfs1_busy;
       wire [`SQ_VTX_STATE_WIDTH-1:0] aarb_state;
                                                                                                                               wire [0:0]
                                                                                                                                                        tif_busy;
       wire [`SQ_STATUS_WIDTH-1:0] aarb_status;
                               aarb_thread_type;
                                acfs0_rtr;
       wire [0:0]
                                                                                                                               wire [0:0]
                                                                                                                                                        aifl_busy;
       wire [0:0]
                                acfsl_rtr;
                                                                                                                               wire [0:0]
                                                                                                                                                        ais0_busy;
11
                                                                                                                       11
                                                                                                                               wire [0:0]
                                                                                                                                                        ais1_busy;
12
       wire [0:0]
                                acfs0_rts;
13
       wire ['SQ_VTX_CTL_PKT_WIDTH-1:0] acfs0_ctl_pkt;
                                                                                                                             wire [0:0] ais0_free_done;
14
       wire [11:0]
                               acfs0_tgt_instr_ptr;
                                                                                                                              wire [0:0] ais0_free_id;
15
       wire [11:0]
                                acfs0 tgt instr cnt;
                                                                                                                       15
                                                                                                                              wire [0:0] ais1 free done;
                                                                                                                               wire [0:0] ais1_free_id;
16
       wire [0:0]
                                aif0 rtr;
                                                                                                                       16
17
                                                                                                                       17
18
       wire [0:0]
                                acfs1 rts;
                                                                                                                       18
19
       wire [`SQ_VTX_CTL_PKT_WIDTH-1:0] acfs1_ctl_pkt;
                                                                                                                       19
                                                                                                                              // -- Vertex Thread Buffer --
20
       wire [11:0]
                                acfs1 tgt instr ptr;
                                                                                                                       20
21
       wire [11:0]
                                acfs1_tgt_instr_cnt;
                                                                                                                       21
22
       wire [0:0]
                                 aifl_rtr;
                                                                                                                       22
23
                                                                                                                       23
                                                                                                                              sq_thread_buff
24
       wire [6:0]
                                param_cache_wptr_q;
                                                                                                                       24
25
                                                                                                                              \label{eq:continuity} $$ \sc vix_state\_width, $$ \sc vix_state\_width, $$ \sc vix_state\_width. $$
                                           Page 33 of 92
                                                                                                                                                                  Page 34 of 92
                                                                               Ex. 2093 - sq.v
                                                                                                                                                                                                      Ex. 2093 - sq.v
 2
                                                                                                                        2
                                                                                                                                // alu control flow seq update of state and status
       u sq vtx thread buff
 3
                                                                                                                                 .acfs0_update (acfs0_update),
                                                                                                                                                                    // load updated status info from CFS
                                              // a strap that tells this module if it's a vertex or
                                                                                                                                 .acfs0 thread type(acfs0 thread type), //
      .thread_type_strap(HI),
pixel thread buffer
     . state\_read\_phase \ (state\_read\_phase), \qquad \textit{// share read access to TB State Mem btwn the tex} \\ and alu arbiters
                                                                                                                                 .acfs0_state (acfs0_state), // state
                                                                                                                                 .acfs0 status (acfs0 status),
                                                                                                                                                                     // status
        .cfs_phase (cfs_phase),
                                      // 00:alu0, 01:tex, 10:alu1, 11:tex
                                                                                                                                 .acfs1 update (acfs1 update),
                                                                                                                                                                          // load updated status info from CFS
        // control packet input (from ISM) - initial values for state and status
                                                                                                                                 .acfs1_thread_type(acfs1_thread_type), //
11
                      (vism_ctl_pkt_rts), // control packet rts
                                                                                                                       10
                                                                                                                                 .acfs1_state (acfs1_state), // state
         .ism_lod_correct (), // state - not used for VTB (PTB only)
                                                                                                                       11
                                                                                                                                 .acfs1_status (acfs1_status), // status
13
         .ism_instr_ptr (vism_instr_ptr), // state
                                                                                                                       12
                                                                                                                                // tex thread arbiter interface
14
         .ism_valid_bits (vism_valid_bits), // state
15
                                                                                                                                .tex_req_q (vtx_tex_req_q), // tex request from every thread in the buffer
         .ism gpr base (vism gpr base), // state
                                                                                                                                .tex_winner_q (vtx_tex_winner_q), // tex winner
16
         .ism context id (vism context id), // state
                                                                                                                                 .tex_winner_ack (vtx_tex_winner_ack), // tex winner valid (request acknowledge)
17
         .ism resource (vism resource), // status: resource bit : tex=1, alu=0
18
         .ism first thread (vism first thread), // status: first thread of a new state
                                                                                                                                .tex_state_q (vtx_tex_state), // winning state read from State Mem
                                           // rtr when not full OR when not doing a CFS
19
20
         .tb_rtr
                         (vtb rtr),
                                                                                                                                .tex_status_q (vtx_tex_status), // winning status read from Status Regs
                                                                                                                       19
      update
                                                                                                                       20
21
                                                                                                                                // done info from TP
22
        // tex control flow seq update of state and status
23
         .tcfs_update (tcfs_update), // load updated status info from CFS
                                                                                                                       22
                                                                                                                                . TP\_SQ\_data\_rdy \; (TP\_SQ\_data\_rdy), \quad \textit{// data ready (done) indicator from TPC}
24
         .tcfs_thread_type (tcfs_thread_type),
                                                                                                                       23
                                                                                                                                 .TP_SQ_type (TP_SQ_type),
                                                                                                                                                                         // the vector type: pixel=0, vertex=1
25
         .tcfs_state (tcfs_state),
                                                                                                                       24
                                                                                                                                 . TP\_SQ\_thread\_id (TP\_SQ\_thread\_id), \  \  /\!/
         .tcfs_status(tcfs_status),
                                                                                                                       25
                                                                                                                                // alu thread arbiter interface
                                           Page 35 of 92
                                                                                                                                                                   Page 36 of 92
                                                                                                                                                                                                       Ex. 2093 - sq.v
                                                                               Ex. 2093 - sq.v
```

```
.alu_req_q (vtx_alu_req_q), // tex request from every thread in the buffer
                                                                                                                                     // outputs from exit SM to constant stores and gpr alloc
2
         .alu_winner_q (vtx_alu_winner_q), // alu winner
                                                                                                                                     // - this needs to be fixed
         .alu winner ack (vtx alu winner ack), // alu winner valid from alu arbiter
                                                                                                                                       .state_change (vtx_state_change), // a pulse high indicates that the state exiting the
         .alu_state_q (vtx_alu_state), // winning state read from State Mem
                                                                                                                                      .old_state (vtx_old_state), // the state that has finished (because a new state has
         .alu_status_q (vtx_alu_status), // winning status read from Status Regs
                                                                                                                                   emerged)
                                                                                                                                      .dealloc_req (vtx_dealloc_req), // request to deallocate GPRs
         // done info from AIS's
                                                                                                                                      .dealloc_ack (vtx_dealloc_ack), // the dealloc request has been acknowleged
                                        // done indicator from AIS0
         .ais0_thread_type (ais0_thread_type), // the vector type: pixel=0 (), vertex=1
                                                                                                                                      .pop thread (vtx vector done), // vtx shader sync output back to pix input ctl
                                                                                                                            10
         .ais0_thread_id (ais0_thread_id), //
                                                                                                                            11
11
                                     // done indicator from AIS1
         .ais1_done (ais1_done),
                                                                                                                                      .param\_cache\_wptr\_q(param\_cache\_wptr\_q), \quad \textit{// from export\_alloc - needed for status reg}
                                                                                                                            12
13
12
         .ais1_thread_type (ais1_thread_type), // the vector type: pixel=0, vertex=1
                                                                                                                            14
13
         .ais1_thread_id (ais1_thread_id), //
                                                                                                                            15
                                                                                                                                      .busy(vtx_tb_busy),
                                                                                                                                                                     // vtx TB busy
                                                                                                                            16
                                                                                                                                      .clk(sclk_global),
15
         // SX export buffer availability
                                                                                                                            17
                                                                                                                                      .reset(srst)
         . u0\_SX\_SQ\_exp\_count\_rdy (u0\_SX\_SQ\_exp\_count\_rdy), \qquad // \ position \ available \ from
                                                                                                                            18
         . u0\_SX\_SQ\_exp\_pos\_avail (u0\_SX\_SQ\_exp\_pos\_avail),
                                                                     // position available from
                                                                                                                            19
                                                                                                                            20
      .u0_SX_SQ_exp_buf_avail(u0_SX_SQ_exp_buf_avail), (0 to 127 2kbit buffers(), 2kbit = 32bits for 64 pixels)
                                                                     // buffer available from SX
                                                                                                                            21
      .ul_SX_SQ_exp_count_rdy(ul_SX_SQ_exp_count_rdy),
                                                                     // position available from
                                                                                                                                    // -- Pixel Thread Buffer --
      .ul_SX_SQ_exp_pos_avail(ul_SX_SQ_exp_pos_avail), SX
                                                                     // position available from
                                                                                                                            25
                                                                                                                                  sq thread buff
      .u1_SX_SQ_exp_buf_avail(u1_SX_SQ_exp_buf_avail), (0 to 127 2kbit buffers(), 2kbit = 32bits for 64 pixels)
                                                                     // buffer available from SX
                                                                                                                            26
28
                                             Page 37 of 92
                                                                                                                                                                          Page 38 of 92
                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                               Ex. 2093 - sq.v
          SQ_PIX_STATE_WIDTH, 'SQ_PIX_ISM_STATE_WIDTH, 'SQ_CFS_STATE_WIDTH,
                                                                                                                                      .tcfs status(tcfs status),
      'SO PIX STATUS WIDTH
 3
       )
                                                                                                                                      // alu control flow seg update of state and status
       u\_sq\_pix\_thread\_buff
                                                                                                                                      .acfs0 update (acfs0 update),
                                                                                                                                                                                  // load updated status info from CFS
 5
                                                                                                                                      .acfs0_thread_type(acfs0_thread_type), //
      .thread_type_strap(LO),
pixel thread buffer
                                             // a strap that tells this module if it's a vertex or
                                                                                                                                      .acfs0_state (acfs0_state), // state
      .state\_read\_phase \ (state\_read\_phase), \qquad /\!/ \ share \ read \ access \ to \ TB \ State \ Mem \ btwn \ the \ tex \ and \ alu \ arbiters
                                                                                                                                      .acfs0_status (acfs0_status), // status
10
         .cfs_phase (cfs_phase),
                                       // 00:alu0, 01:tex, 10:alu1, 11:tex
                                                                                                                                      .acfs1_update (acfs1_update),
                                                                                                                                                                                 // load updated status info from CFS
11
                                                                                                                            10
                                                                                                                                      .acfsl_thread_type(acfsl_thread_type), //
12
        // control packet input (from ISM) - initial values for state and status
                                                                                                                            11
                                                                                                                                      .acfsl_state (acfsl_state),
13
                       (pism ctl pkt rts), // control packet rts
                                                                                                                            12
                                                                                                                                      .acfs1_status (acfs1_status),
14
         .ism lod correct (pism lod correct). // state
                                                                                                                            13
15
         .ism instr ptr (pism instr ptr). // state
         .ism_valid_bits (pism_valid_bits), // state
16
                                                                                                                                      .tex_req_q (pix_tex_req_q), // tex request from every thread in the buffer
17
         .ism_gpr_base (pism_gpr_base), // state
                                                                                                                                      .tex_winner_q (pix_tex_winner_q), // tex winner
18
         .ism_context_id (pb_pix_state),
                                                     // state
                                                                                                                                       .tex_winner_ack (pix_tex_winner_ack), // tex winner valid (request acknowledge)
         .ism_resource (pism_resource), // status: resource bit : tex=1, alu=0
19
                                                                                                                            19
20
         .ism_first_thread (pism_first_thread), // status: first thread of a new state
                                                                                                                                      .tex_state_q (pix_tex_state), // winning state read from State Mem
                                                                                                                            20
                         (ptb_rtr),
                                               // rtr when not full OR when not doing a CFS
                                                                                                                                      .tex\_status\_q \qquad (pix\_tex\_status), \quad // \ winning \ status \ read \ from \ Status \ Regs
                                                                                                                            21
23
                                                                                                                            22
                                                                                                                                     // done info from TP
24
         // tex control flow seq update of state and status
                                                                                                                            23
                                                                                                                                      . TP\_SQ\_data\_rdy \; (TP\_SQ\_data\_rdy), \quad \textit{// data ready (done) indicator from TPC}
25
         .tcfs_update (tcfs_update), // load updated status info from CFS
                                                                                                                            24
                                                                                                                                      .TP_SQ_type (TP_SQ_type),
                                                                                                                                                                                  // the vector type: pixel=0, vertex=1
26
         .tcfs_thread_type (tcfs_thread_type),
                                                                                                                            25
                                                                                                                                      . TP\_SQ\_thread\_id (TP\_SQ\_thread\_id), \  \  /\!/
         .tcfs_state (tcfs_state),
                                                                                                                            26
                                            Page 39 of 92
                                                                                                                                                                          Page 40 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                               Ex. 2093 - sq.v
```

```
1
        // alu thread arbiter interface
2
         .alu_req_q (pix_alu_req_q), \hspace{0.2in} // alu req from every thread
                                                                                                                            2
                                                                                                                                    // outputs from exit SM to constant stores and gpr alloc
         .alu\_winner\_q \qquad (pix\_alu\_winner\_q), \ \ /\!/ \ alu \ winner
                                                                                                                                    // - this needs to be fixed...
         .alu_winner_ack (pix_alu_winner_ack), // alu winner valid from alu arbiter
                                                                                                                                .alu_state_q (pix_alu_state), // winning state read from State Mem
                                                                                                                                  .old_state (pix_old_state), \ \ /\!\!/ the state that has finished (because a new state has emerged)
          .alu_status_q (pix_alu_status), // winning status read from Status Regs
                                                                                                                                    .dealloc req (pix dealloc req), // request to deallocate GPRs
         // done info from AIS's
                                                                                                                                    .dealloc_ack (pix_dealloc_ack), // the dealloc request has been acknowleged
                                       // done indicator from AIS0
         . ais0\_thread\_type \ (ais0\_thread\_type), \quad \textit{// the vector type: pixel=0} \qquad (), \, vertex=l
                                                                                                                           11
                                                                                                                                    .pop thread (), // no connect for pix TB
11
         .ais0_thread_id (ais0_thread_id), //
                                                                                                                           12
12
         .ais1_done (ais1_done),
                                       // done indicator from AIS1
                                                                                                                                     .param\_cache\_wptr\_q(param\_cache\_wptr\_q), \quad \textit{// from export\_alloc - needed for status reg}
                                                                                                                            14 request logic
13
         .aisl_thread_type (aisl_thread_type), // the vector type: pixel=0, vertex=1
                                                                                                                                                                          // not used by pix TB since type must be VTX in
         .ais1_thread_id (ais1_thread_id), //
                                                                                                                                request logic
15
                                                                                                                           17
16
         // SX export buffer availability
                                                                                                                           18
                                                                                                                                     .busy(pix_tb_busy), // pix TB busy
         . u0\_SX\_SQ\_exp\_count\_rdy (u0\_SX\_SQ\_exp\_count\_rdy), \qquad // \ position \ available \ from
                                                                                                                                    .clk(sclk_global),
                                                                                                                           19
                                                                                                                           20
                                                                                                                                    .reset(srst)
      .u0_SX_SQ_exp_pos_avail(u0_SX_SQ_exp_pos_avail),
                                                                    // position available from
                                                                                                                           21
      .u0_SX_SQ_exp_buf_avail(u0_SX_SQ_exp_buf_avail),
(0 to 127 2kbit buffers(), 2kbit = 32bits for 64 pixels)
                                                                    // buffer available from SX
      .u1\_SX\_SQ\_exp\_count\_rdy(u1\_SX\_SQ\_exp\_count\_rdy),\\SX
                                                                    // position available from
                                                                                                                           24
      .ul_SX_SQ_exp_pos_avail(ul_SX_SQ_exp_pos_avail), SX
                                                                                                                           25
                                                                                                                                  // -- Texture Thread Arbiter --
                                                                    // position available from
                                                                                                                           26
     .ul_SX_SQ_exp_buf_avail(ul_SX_SQ_exp_buf_avail), (0 to 127 2kbit buffers(), 2kbit = 32bits for 64 pixels)
                                                                    // buffer available from SX
                                            Page 41 of 92
                                                                                                                                                                        Page 42 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                             Ex. 2093 - sq.v
       sq_thread_arb
                                                                                                                                     .arb_rts0
                                                                                                                                                     (tarb_rts),
                                                                                                                                                                          // ready to send the winner to CFS0
2
                                                                                                                            2
                                                                                                                                                     0.
                                                                                                                                     .arb rts1
                                                                                                                                                                        // no connect for tex thread arb
        `SQ_STATE_WIDTH, `SQ_STATUS_WIDTH
3
                                                                                                                                     .arb state (tarb state).
                                                                                                                                                                   // the state sent to the CFS
4
                                                                                                                                                                   // the status sent to the CFS
                                                                                                                                     .arb status (tarb status).
5
       u\_sq\_tex\_thread\_arb
                                                                                                                                     .arb\_thread\_type \quad (tarb\_thread\_type), \quad \  \  /\!/ \ vtx \ or \ pix
6
7
         .arb\_type\_strap \hspace{0.5cm} (HI), \hspace{0.5cm} /\!/ tex = 1, \, alu = 0
                                                                                                                                     .cfs rtr0
                                                                                                                                                      (tefs rtr).
                                                                                                                                                                           // CFS0 can accept a thread
          .state_read_phase (state_read_phase), // share read access to TB State Mem btwn the tex
                                                                                                                                     .cfs rtr1
                                                                                                                                                      (LO).
                                                                                                                                                                           // always tied LO for tex thread arb
10
                                                                                                                                    .cfs1 enable
                                                                                                                                                                           // always tied LO for tex thread arb
                                                                                                                           10
                                                                                                                                                    (LO),
11
         // vertex and pixel thread buffer interface
                                                                                                                           11
         .vtx\_req\_q \, (vtx\_tex\_req\_q), \qquad /\!/ \, 16 \, vtx\_thread\_buff \, requests
12
                                                                                                                           12
                                                                                                                                    .clk(sclk_global),
      13
15
         .vtx winner ack (vtx tex winner ack),
                                                                                                                           15
         .vtx state (vtx tex state), // state selected by winner
16
         .vtx_status (vtx_tex_status), // status selected by winner
17
18
         .pix\_req\_q \ (pix\_tex\_req\_q), \qquad \  \  /\!\!/ \ 16 \ pix\_thread\_buff \ requests
                                                                                                                           18
                                                                                                                                  // -- Tex CFS --
19
20 __pix_winner_q _ (pix_tex_winner_q), // winning pixel thread_id sent back to Pixel 21 __Thread_Buffer
                                                                                                                           20
22
         .pix_winner_ack (pix_tex_winner_ack),
                                                                                                                           21
                                                                                                                                  sq ctl flow seq
23
                                                                                                                           22
24
         .pix_status (pix_tex_status),
                                                                                                                           23
                                                                                                                                    `SQ_CTL_PKT_WIDTH, `SQ_STATE_WIDTH, `SQ_STATUS_WIDTH
25
                                                                                                                           24
        // control flow sequencer interface
                                                                                                                                   u\_sq\_tex\_ctl\_flow\_seq
                                             Page 43 of 92
                                                                                                                                                                        Page 44 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                             Ex. 2093 - sq.v
```

```
1
2
        .cfs_type_strap (2'b01),
                                          // 00:alu0, 01:tex, 10:alu1
                                                                                                                                 // interface to the thread buffer (for thread updates)
         . is\_phase \qquad \qquad (is\_phase), \qquad \qquad /\!/\ 00: CF, \ 01: Tex, \ 10: ALU, \ 11: CP
                                                                                                                                 .cfs\_update\_q \qquad (tcfs\_update), \qquad \qquad //\ load\ updated\ status\ info\ from\ CFS
         .is_subphase (is_subphase), // 00:alu0, 01:tex, 10:alu1, 11:tex
                                                                                                                                 .cfs_state (tcfs_state), // state
         .cfs_phase (cfs_phase), // 00:alu0, 01:tex, 10:alu1, 11:tex
                                                                                                                                 .cfs_status (tcfs_status), // status
        // local registers
                                                                                                                                 // outputs to the target instruction fetcher
                                                                                                                                 .cfs_rts_q (tcfs_rts), // ctl packet and ptr are valid
                                                                                                                            .cfs_ctl_pkt_q (tcfs_ctl_pkt), // the control packet (lod_correct, valid_bits, gpr_base, context_id)
        .inst_base_vtx (inst_base_vtx), // vertex base (wrap point)
        .inst_base_pix (inst_base_pix), // pixel base (wrap point)
                                                                                                                                  .cfs\_tgt\_instr\_ptr\_q(tcfs\_tgt\_instr\_ptr), \ /\!/ \ the \ instr \ store \ address \ of \ the \ first \ target
11
     .vs_program_base_set(vs_program_base_set),// connected to SQ_VS_PROGRAM.BASE (12 bits)
12
13
                                                                                                                        13
                                                                                                                                 .cfs_tgt_instr_cnt_q(tcfs_tgt_instr_cnt), // the number of target instructions to be fetched
                                                                                                                        14
                                                                                                                                 .cfs_thread_type_q(tcfs_thread_type), // vertex or pixel
14
15
     .ps_program_base_set(ps_program_base_set),// connected to SQ_PS_PROGRAM.BASE (12 bits)
                                                                                                                        15
                                                                                                                                 //.cfs_param_ptr_q(), // param cache ptr - not needed for texture
16
                                                                                                                        16
                                                                                                                                 .tif_rtr (tif_rtr),
                                                                                                                                                                   // TIF can take a new packet
17
        // thread arbiter interface
                                                                                                                        17
18
         .arb_rts (tarb_rts),
                                                                                                                        18
                                                                                                                                 .busy(tcfs_busy),
19
                                                                                                                        19
                                                                                                                                  .clk(sclk_global),
20
                                                                                                                        20
                                                                                                                                 .reset(srst)
         .arb_status (tarb_status),
21
         .arb_thread_type (tarb_thread_type), // vertex or pixel
                                                                                                                        21
22
                                   // CFS can take a new packet
                                                                                                                        22
         .cfs_rtr_q (tcfs_rtr),
23
                                                                                                                        23
24
25
         .is_read_addr_q (tcfs_is_read_addr), // instruction store read address
                                                                                                                               // -- ALU Thread Arbiter --
         .is_read_data_q (is_read_data),
                                           // instruction store read data
                                           Page 45 of 92
                                                                                                                                                                    Page 46 of 92
                                                                                Ex. 2093 - sq.v
                                                                                                                                                                                                        Ex. 2093 - sq.v
                                                                                                                                 // control flow sequencer interface
2
                                                                                                                         2
                                                                                                                                              (aarb rts0),
      sq thread arb
                                                                                                                                 .arb rts0
                                                                                                                                                                      // ready to send the winner to CFS0
3
                                                                                                                                  .arb rts1
                                                                                                                                              (aarb_rts1),
                                                                                                                                                                      // ready to send the winner to CFS1
4
       `SQ_VTX_STATE_WIDTH, `SQ_STATUS_WIDTH
                                                                                                                                  .arb_state (aarb_state), // the state sent to the CFS
5
                                                                                                                                 .arb_status (aarb_status), // the status sent to the CFS
 6
      u_sq_alu_thread_arb
                                                                                                                                 .arb_thread_type (aarb_thread_type), // vtx or pix
7
                                                                                                                                  .cfs\_rtr0 \hspace{1.5cm} (acfs0\_rtr), \hspace{1.5cm} /\!/ \, CFS0 \ can \ accept \ a \ thread
                                                                                                                                  .cfs_rtrl (acfsl_rtr),
8
         .arb type strap (LO), // \tan = 1, alu = 0
                                                                                                                                                                      // CFS1 can accept a thread
     .state_read_phase (state_read_phase), // share read access to TB State Mem btwn the tex and alu arbiters
                                                                                                                        10
                                                                                                                                 //.cfs1_enable (LO),
                                                                                                                                                                      // enable sending packets to CFS1
11
                                                                                                                        11
                                                                                                                                 .cfs1_enable (HI),
                                                                                                                                                                      // enable sending packets to CFS1
12
        // vertex and pixel thread buffer interface
                                                                                                                        12
                                                                                                                                                  // (this a local register setting: SQ_FLOW_CTL.ONE_ALU)
13
        .vtx_req_q (vtx_alu_req_q), // 16 vtx_thread_buff requests
                                                                                                                                 .clk(sclk_global),
          .vtx_winner_q (vtx_alu_winner_q), // winning vertex thread_id sent back to Vertex
                                                                                                                        14
                                                                                                                                 .reset(srst)
                                                                                                                        15
16
        .vtx winner ack (vtx alu winner ack),
        .vtx_state (vtx_alu_state), // state selected by winner
17
        .vtx_status (vtx_alu_status), // status selected by winner
18
                                                                                                                        18
19
                                                                                                                             // -- Export Alloc --
20
         .pix\_req\_q \, (pix\_alu\_req\_q), \qquad \textit{// 16 pix\_thread\_buff requests}
21 __pix_winner_q (pix_alu_winner_q), // winning pixel thread_id sent back to Pixel 22 __ThreadBuffer
                                                                                                                        20
                                                                                                                        21
23
         .pix_winner_ack (pix_alu_winner_ack),
                                                                                                                        22
                                                                                                                              sq export alloc
24
        .pix_state (pix_alu_state), //
                                                                                                                        23
                                                                                                                                u sq export alloc
25
         .pix_status (pix_alu_status), //
                                                                                                                        24 (
26
                                                                                                                        25
                                                                                                                               // inputs from local registers
                                            Page 47 of 92
                                                                                                                                                                    Page 48 of 92
                                                                                Ex. 2093 - sq.v
                                                                                                                                                                                                        Ex. 2093 - sq.v
```

```
.vs_export_count_set (vs_export_count_set),
SQ_PROGRAM_CNTL.VS_EXPORT_COUNT (4 bits)
                                                                        connected
                                                                                            to
                                                                                                                                 .SQ_SX_exp_context_id (SQ_SX_exp_state),
                                                                                                                         2
                                                                                                                                 .SQ_SX_exp_id
                                                                                                                                                      (SQ_SX_exp_id),
     .vs_export_mode_set (vs_export_mode_set),
SQ_PROGRAM_CNTL.VS_EXPORT_MODE (3 bits)
                                                                        connected
     .ps_export_mode_set (ps_export_mode_set), SQ_PROGRAM_CNTL.PS_EXPORT_MODE (3 bits)
                                                                                                                                 .ais0_free_done
                                                                                                                                                        (ais0_free_done),
                                                                                                                                 .ais0_free_id
                                                                                                                                                        (ais0_free_id),
                                                                                                                                  .ais1_free_done
                                                                                                                                                        (ais1_free_done),
         .alu arb rts0
                               (aarb rts0),
                                                    // ready to send the winner to CFS0
                                                                                                                                                        (ais1_free_id),
         .alu arb rts1
                               (aarb rts1).
                                                    // ready to send the winner to CFS1
         .alu arb context id
                                                                                                                                                             (SQ_SX_free_done),
                                                                                                                                 .SQ_SX_free_done
           (aarb_state['SQ_CFS_STATE_WIDTH+2:'SQ_CFS_STATE_WIDTH]),
                                                                                                                                 .SQ_SX_free_id
                                                                                                                                                        (SQ_SX_free_id),
12
         .alu arb status
                                (aarb status), // the status sent to the CFS
13
         .alu_arb_thread_type (aarb_thread_type), // vtx or pix
                                                                                                                        12
                                                                                                                                 // - export id interface
14
                                                                                                                                  .cfs0_export_id (acfs0_state[7]),
                                                                                                                                                                      // export id that cfs0 is pushing down pipe 0 (sets
15
         .alu0_cfs_rtr
                                 (acfs0_rtr),
                                                     // ALU_CFS0 can accept a thread
         .alu1_cfs_rtr
                                 (acfsl_rtr),
                                                    // ALU_CFS1 can accept a thread (for alu
                                                                                                                        15
                                                                                                                                 .cfs_aif_xfc0 (cfs_aif_xfc0),
                                                                                                                                                                      // cfs0 to aif0 transfer complete
                                                                                                                              .cfs1_export_id (acfs1_state[7]), global export_id)
18
                                                                                                                                                                      // export_id that cfs1 is pushing down pipe 1 (sets
19
        .pb_dealloc_cnt
                                (pb_dealloc_cnt), // param cache dealloc info
                                                                                                                        18
                                                                                                                                 .cfs_aif_xfcl (cfs_aif_xfcl),
                                                                                                                                                                      // cfs1 to aif1 transfer complete
20
         .pb_dealloc_vld
                                (pb_dealloc_vld),
                                                                                                                        19
21
         .param cache wptr q (param cache wptr q),
                                                                                                                        20
                                                                                                                                  .global\_export\_id\_q(global\_export\_id), \  \  // \  exp\_id \  output \  to \  ALU \  CFS's
22
                                                                                                                        21
23
        //.SQ SX exp pix
                                      (SQ SX exp pix),
                                                                                                                        22
                                                                                                                                 .clk(sclk_global),
24
                                                                                                                        23
25
        .SQ SX exp valid
                                      (SQ SX exp valid),
                                                                                                                        24
26
        .SQ_SX_exp_type
                               (SQ_SX_exp_type),
                                                                                                                        25
2.7
        .SQ_SX_exp_number
                                       (SQ_SX_exp_number),
                                            Page 49 of 92
                                                                                                                                                                    Page 50 of 92
                                                                                Ex. 2093 - sq.v
                                                                                                                                                                                                        Ex. 2093 - sq.v
                                                                                                                                 // thread arbiter interface
      // -- ALU CFS 0 --
                                                                                                                                 .arb_rts (aarb_rts0),
2
                                                                                                                         2
                                                                                                                                  .arb state (aarb state),
4
                                                                                                                                 .arb status (aarb status),
5
      sq_ctl_flow_seq
                                                                                                                                  .arb_thread_type (aarb_thread_type), // vertex or pixel
6
                                                                                                                                 .cfs_rtr_q (acfs0_rtr), // CFS can take a new packet
7
        'SQ VTX CTL PKT WIDTH, 'SQ VTX STATE WIDTH, 'SQ STATUS WIDTH
8
                                                                                                                                  .pc\_base\_q(param\_cache\_wptr\_q), \qquad \textit{// from sq\_export\_alloc}
       u0_sq_alu_ctl_flow_seq
                                                                                                                                 // instruction store interface
10
                                                                                                                        10
11
         .cfs_type_strap (2'b00),
                                           // 00:alu0, 01:tex, 10:alu1
                                                                                                                        11
                                                                                                                                 .is_read_addr_q \, (acfs0_is_read_addr), // instruction store read address
12
                                                                                                                        12
                                                                                                                                 .is_read_data_q (is_read_data), // instruction store read data
13
                                           // 00:CF, 01:Tex, 10:ALU, 11:CP
                                                                                                                        13
                      (is_phase),
         .is_subphase (is_subphase), // 00:alu0, 01:tex, 10:alu1, 11:tex
                                                                                                                                 // interface to the thread buffer (for thread updates)
15
         .cfs_phase (cfs_phase),
                                  // 00:alu0, 01:tex, 10:alu1, 11:tex
                                                                                                                                 .cfs_update_q (acfs0_update), // load updated status info from CFS
16
                                                                                                                                 .cfs_state (acfs0_state), // state
17
        // local registers
                                                                                                                                 .cfs_status (acfs0_status), // status
18
        // - per chip
19
        .inst base vtx (inst base vtx), // vertex base (wrap point)
                                                                                                                                // outputs to the target instruction fetcher
                                                                                                                                                             // ctl packet and ptr are valid
                                                                                                                        20
20
        .inst_base_pix (inst_base_pix),
                                             // pixel base (wrap point)
                                                                                                                                 .cfs rts q (acfs0 rts),
21
        // - per context
                                                                                                                                  .cfs_ctl_pkt_q (acfs0_ctl_pkt),
                                                                                                                                                                          // the control packet (lod_correct,
                                                                                                                            valid_bits, gpr_base, context_id)
     .vs_program_base_set(vs_program_base_set),// connected to SQ_VS_PROGRAM.BASE (12 bits)
                                                                                                                                 .cfs_tgt_instr_ptr_q(acfs0_tgt_instr_ptr),// the instr store address of the first target
     .ps_program_base_set(ps_program_base_set),// connected to SQ_PS_PROGRAM.BASE (12 bits)
                                                                                                                             .cfs\_tgt\_instr\_cnt\_q(acfs0\_tgt\_instr\_cnt),\\ fetched
                                                                                                                                                                          // the number of target instructions to be
26
                                            Page 51 of 92
                                                                                                                                                                    Page 52 of 92
                                                                               Ex. 2093 - sq.v
                                                                                                                                                                                                        Ex. 2093 - sq.v
```

```
// 00:alu0, 01:tex, 10:alu1. 11:tex
 1
          .cfs_thread_type_q(acfs0_thread_type), // vertex or pixel
                                                                                                                                      .is subphase (is subphase),
 2
          .tif rtr (aif0 rtr).
                                             // TIF can take a new packet
                                                                                                                                      .cfs_phase (cfs_phase), // 00:alu0, 01:tex, 10:alu1, 11:tex
          . global\_export\_id \ (global\_export\_id), \quad \textit{//} \ exp\_id \ input \ from \ sq\_exp\_alloc
                                                                                                                                      // local registers
          .cfs_tif_xfc (cfs_aif_xfc0), // cfs0 to aif0 transfer complete (to sq_exp_alloc)
                                                                                                                                      .inst_base_vtx (inst_base_vtx), // vertex base (wrap point)
          .busy(acfs0_busy),
                                                                                                                                      .inst_base_pix (inst_base_pix), // pixel base (wrap point)
          .clk(sclk_global),
                                                                                                                                   .vs_program_base_set(vs_program_base_set),// connected to SQ_VS_PROGRAM.BASE (12 bits)
10
                                                                                                                                   .ps_program_base_set(ps_program_base_set),// connected to SQ_PS_PROGRAM.BASE (12 bits)
11
12
                                                                                                                             13
13
                                                                                                                             14
                                                                                                                                      // thread arbiter interface
       // -- ALU CFS 1 --
                                                                                                                             15
                                                                                                                                      .arb rts (aarb rts1).
15
                                                                                                                             16
                                                                                                                                      .arb_state (aarb_state),
16
                                                                                                                             17
                                                                                                                                      .arb_status (aarb_status),
17
       sq_ctl_flow_seq
                                                                                                                             18
                                                                                                                                       .arb_thread_type (aarb_thread_type), // vertex or pixel
18
                                                                                                                             19
                                                                                                                                       .cfs_rtr_q (acfs1_rtr), // CFS can take a new packet
19
        'SO VTX CTL PKT WIDTH, 'SO VTX STATE WIDTH, 'SO STATUS WIDTH
                                                                                                                             20
20
                                                                                                                             21
                                                                                                                                      .pc_base_q(param_cache_wptr_q), // from sq_export_alloc
21
       ul_sq_alu_ctl_flow_seq
                                                                                                                             22
22
                                                                                                                             23
23
         .cfs_type_strap (2'b10),
                                             // 00:alu0, 01:tex, 10:alu1
                                                                                                                             24
                                                                                                                                      .is_read_addr_q (acfs1_is_read_addr), // instruction store read address
24
                                                                                                                             25
                                                                                                                                      .is_read_data_q (is_read_data), // instruction store read data
25
          .is_phase
                       (is_phase),
                                              // 00:CF, 01:Tex, 10:ALU, 11:CP
                                             Page 53 of 92
                                                                                                                                                                           Page 54 of 92
                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                                Ex. 2093 - sq.v
         // interface to the thread buffer (for thread updates)
 2
         .cfs_update_q (acfs1_update),
                                                     // load updated status info from CFS
                                                                                                                                    // -- Texture Instruction Pipe --
          .cfs_state (acfs1_state), // state
          .cfs status (acfs1 status), // status
         // outputs to the target instruction fetcher
                                                                                                                              6 // - interconnect wires
         .cfs_rts_q (acfs1_rts),
                                    // ctl packet and ptr are valid
      .cfs\_ctl\_pkt\_q \quad (acfsl\_ctl\_pkt), \qquad \qquad // \quad the \quad control \quad packet \quad (lod\_correct, valid\_bits, gpr\_base, context\_id)
                                                                                                                                    wire [0:0] alu phase;
      .cfs\_tgt\_instr\_ptr\_q(acfs1\_tgt\_instr\_ptr), /\!/ \  \  the \  \  instr \  \  store \  \  address \  \  of \  \  the \  \  first \  \  target instruction
                                                                                                                                  wire [04:0] texconst_rd_addr;
32 consts)
                                                                                                                                                                    // texture constant store read address (logical addr - up to
      .cfs\_tgt\_instr\_cnt\_q(acfs1\_tgt\_instr\_cnt), \hspace{1cm} \textit{// the number of target instructions to be fetched} \\
                                                                                                                                    wire [95:0] texconst_rd_data; // texture constant store read data
14
          .cfs_thread_type_q(acfs1_thread_type), // vertex or pixel
                                                                                                                                    wire [0:0] tif_instr_rts; // the target instr register is valid
15
                                             // TIF can take a new packet
                        (aif1 rtr).
                                                                                                                                    wire [95:0] tif_instr;// the target instruction register (TIR)
16
                                                                                                                                     wire [`SQ_CTL_PKT_WIDTH-1:0] tif_ctl_pkt; // the target control packet (pipelined from
17
          .global_export_id (global_export_id), // exp_id input from sq_exp_alloc
         .cfs_tif_xfc (cfs_aif_xfc1), // cfs0 to aif0 transfer complete (to sq_exp_alloc)
18
                                                                                                                                    wire [0:0] tif_last_in_group; // last instruction flag
                                                                                                                             18
19
                                                                                                                                     wire [0:0] tif_thread_type; // vert:1, pix:0
                                                                                                                             19
20
          .busy(acfs1 busy).
                                                                                                                                     wire [5:0] tif_thread_id; // the target thread_id (pipelined from reg'd input)
                                                                                                                             20
21
          .clk(sclk_global),
                                                                                                                             21
22
          .reset(srst)
                                                                                                                             22
                                                                                                                                    wire [00:0] tiq_rtr;
23
                                                                                                                             23
24
                                                                                                                             24
                                                                                                                                     wire [02:0] tiq_context_id;
25
                                                                                                                             25
                                                                                                                                     wire [63:0] tiq_valid_bits;
                                                                                                                                     wire [95:0] tiq_lod_correct;
                                             Page 55 of 92
                                                                                                                                                                           Page 56 of 92
                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                                Ex. 2093 - sq.v
```

```
wire [00:0] tiq_thread_type;
 2
            wire [05:0] tiq_thread_id;
                                                                                                                                                                                                       2
                                                                                                                                                                                                                   // cfs interface
 3
            wire [95:0] tiq_instr;
                                                                                                                                                                                                                   .cfs rts
                                                                                                                                                                                                                                              (tcfs rts).
                                                                                                                                                                                                                                                                                // ctl packet and ptr are valid
                                                                                                                                                                                                              .cfs_ctl_pkt (tcfs_ctl_pkt),
gpr_base, context_id)
                                                                                                                                                                                                                                                                                // the control packet (lod for pix_tex, valid_bits,
            wire [0:0] tis_rtr;
                                                                                                                                                                                                              .cfs_instr_ptr $(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction <math display="inline">$(tcfs\_tgt\_instr\_ptr),$\ //\ the Instruction Store address of the first target instruction Store 
            wire [6:0] tis_gpr_rd_addr; // GPR read address for Fetch Address
                                                                                                                                                                                                                   .cfs_instr_cnt (tcfs_tgt_instr_cnt), // the number of instructions to be fetched
                                                                                                                                                                                                                   .cfs_pc_base (tcfs_state[7:1]), // the param cache base - not used by tex instr
            wire [0:0] tiq_rts;
                                                                                                                                                                                                     11
                                                                                                                                                                                                                     .cfs_thread_type (tcfs_thread_type), // vertex or pixel
 10
                                                                                                                                                                                                     12
                                                                                                                                                                                                                    .cfs thread id (tcfs status[21:16]), //
11
                                                                                                                                                                                                     13
                                                                                                                                                                                                                    .cfs_last_in_thread(tcfs_status[12]), // last instr in shader prog
12
           // -- Texture Instruction Fetch --
                                                                                                                                                                                                     14
                                                                                                                                                                                                                                              (tif_rtr),
                                                                                                                                                                                                                                                                                // TIF can take a new packet
13
                                                                                                                                                                                                     15
14
                                                                                                                                                                                                     16
                                                                                                                                                                                                                   // instruction store interface
15
            sq target instr fetch
                                                                                                                                                                                                     17
                                                                                                                                                                                                                     . is\_read\_addr \qquad (tif\_is\_read\_addr), \qquad \textit{// instruction store read address}
16
                                                                                                                                                                                                     18
                                                                                                                                                                                                                     .is_read_data (is_read_data), // instruction store read data
17
             `SQ_CTL_PKT_WIDTH // tex pipe needs LOD bits from PIX ctl pkts
                                                                                                                                                                                                     19
                                                                                                                                                                                                                     .is_phase (is_phase),
                                                                                                                                                                                                                                                                               // instruction store phase
18
                                                                                                                                                                                                             19
            u sq tex instr fetch
20
          .target_strap(`SQ_TEX_STRAP),
ALU0_STRAP, or ALU1_STRAP
                                                                                       // hardwired to TEX STRAP.
                                                                                                                                                                                                                   // outputs to the target instruction decoder
 22
                                                                                                                                                                                                                     .tif_pc_base_q (), // the param cache base output - not connected to
23
24
                                                                                                                                                                                                             .tif_ctl_pkt_q (tif_ctl_pkt), // the target control packet (pipelined from reg'd input)
25
                .inst_base_vtx(inst_base_vtx),
                                                                                                                                                                                                                    .tif\_last\_in\_group\_q(tif\_last\_in\_group), \ //\ last\ instruction\ flag
                .inst_base_pix(inst_base_pix),
                                                                            // pixel base
                                                                        Page 57 of 92
                                                                                                                                                                                                                                                                             Page 58 of 92
                                                                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                                                                                                                                                        Ex. 2093 - sq.v
               .tif\_thread\_type\_q(tif\_thread\_type), \qquad \textit{// vert:1, pix:0}
  2
               . tif\_thread\_id\_q \quad (tif\_thread\_id), \qquad \qquad /\!/ \ the \ target \ thread\_id \ (pipelined \ from \ reg'd \ input)
                                                                                                                                                                                                       2
                                                                                                                                                                                                                   // queue inputs
                                                     // the target instruction register (TIR)
                .tif instr a(tif instr).
                                                                                                                                                                                                                    //write data().
 4
                .tif_instr_rts_q (tif_instr_rts), // the target instr register is valid
                                                                                                                                                                                                                     .tif\_ctl\_pkt\_q \qquad (tif\_ctl\_pkt),
                                                                                                                                                                                                                                                                                 // control packet
          .tiq_rtr (tiq_rtr),
(and other pipeline data)
                                                                       // the target instr decode is ready to take the TIR
                                                                                                                                                                                                                     .tif\_thread\_id\_q \quad (tif\_thread\_id),
                                                                                                                                                                                                                                                                                // thread id
                                                                                                                                                                                                                     .tif_last_instr_q (tif_last_in_group), // last instruction flag
                                                                                                                                                                                                                     .tif\_thread\_type\_q(tif\_thread\_type), \qquad \textit{// 0: pixel, 1: vertex}
                .busy(tif_busy),
                                                                                                                                                                                                                                               (tif instr),
                                                                                                                                                                                                                                                                                 // instruction register (TIR)
                .clk(sclk_global),
 10
                                                                                                                                                                                                      10
                                                                                                                                                                                                                     .read rts
                                                                                                                                                                                                                                                (tiq_rts),
 11
                                                                                                                                                                                                     11
 12
                                                                                                                                                                                                     12
                                                                                                                                                                                                                    //read_data(),
                                                                                                                                                                                                                                                // {control packet, clause num, instruction}
 13
                                                                                                                                                                                                     13
                                                                                                                                                                                                                    // queue outputs
15
            // -- Texture Instruction Queue --
                                                                                                                                                                                                                     .tiq_last_instr (tiq_last_instr), // last instruction flag
 16
                                                                                                                                                                                                                     .tiq_thread_type (tiq_thread_type), // 0: pixel, 1: vertex
17
                                                                                                                                                                                                                     .tiq_context_id (tiq_context_id), // context_id (from ctl packet)
18
            parameter TIQ_NUM_WORDS = 4;
                                                                                                                                                                                                                     .tiq_valid_bits (tiq_valid_bits), // valid bits (from ctl packet)
19
            parameter TIQ ADDR BITS = 2;
                                                                                                                                                                                                                     .tiq_lod_correct (tiq_lod_correct), // lod_correct bits (from ctl packet)
20
                                                                                                                                                                                                                    .tiq_thread_id (tiq_thread_id), // thread_id
                                                                                                                                                                                                     20
21
            sq tex instr queue
                                                                                                                                                                                                     21
                                                                                                                                                                                                                                                             // instruction
                                                                                                                                                                                                                    .tiq instr (tiq instr),
22
           // #(TIQ_NUM_WORDS, TIQ_ADDR_BITS)
23
            u\_sq\_tex\_instr\_queue
                                                                                                                                                                                                     23
                                                                                                                                                                                                                    .clk(sclk_global),
24
                                                                                                                                                                                                     24
                                                                                                                                                                                                                    reset(srst)
25
               .write_rts (tif_instr_rts),
                                                                                                                                                                                                     25
                .write_rtr (tiq_rtr),
                                                                        Page 59 of 92
                                                                                                                                                                                                                                                                             Page 60 of 92
                                                                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                                                                                                                                                        Ex. 2093 - sq.v
```

```
.texconst rd addr (texconst rd addr), // texture constant store read address
 2
                                                                                                                                 .texconst rd data (texconst rd data), // texture constant store read data
      // -- Texture Instruction Sequencer --
                                                                                                                                // outputs to TP
                                                                                                                                 .SQ_TP_vld (SQ_TP_send),
                                                                                                                                 .SQ_TP_instr (SQ_TP_instr),
                                                                                                                                 .SQ_TP_const (SQ_TP_const),
       sq_tex_instr_seq
                                                                                                                                 .SQ_TP_gpr_phase (SQ_TP_gpr_phase),
       u_sq_tex_instr_seq
                                                                                                                            . SQ\_TP\_gpr\_wr\_addr (SQ\_TP\_gpr\_wr\_addr), \quad /\!/ \  \  sends \  \  gpr\_wr\_addr \  \  plus \  \  type \  \  over \  \  4 \  \  cycles
10
        // TIQ interface
                                                                                                                       11
                                                                                                                                 .SQ_TP_thread_id(SQ_TP_thread_id), // sends thread_id plus end_of_group over 4 cycles
11
        .tiq_rts (tiq_rts),
                                              // rts from TIQ FIFO
                                                                                                                       12
12
         .tiq_last_instr (tiq_last_instr), // last instruction flag
                                                                                                                       13
                                                                                                                                 .u0 SQ TP lod correct(u0 SQ TP lod correct),
13
         .tiq_thread_type (tiq_thread_type), // 0: pixel, 1: vertex
                                                                                                                       14
                                                                                                                                 .ul SQ TP lod correct(ul SQ TP lod correct),
14
         .tiq_context_id (tiq_context_id),
                                             // context_id (from ctl packet)
                                                                                                                       15
                                                                                                                                 .u2 SQ TP lod correct(u2 SQ TP lod correct),
15
         .tiq_valid_bits (tiq_valid_bits), // valid bits (from ctl packet)
                                                                                                                       16
                                                                                                                                 .u3 SQ TP lod correct(u3 SQ TP lod correct),
16
         .tiq_lod_correct (tiq_lod_correct), // lod_correct bits (from ctl packet)
                                                                                                                       17
17
         .tiq_thread_id (tiq_thread_id), // thread_id
                                                                                                                       18
                                                                                                                                .u0\_SQ\_TP\_pix\_mask(u0\_SQ\_TP\_pix\_mask),
18
         .tiq_instr (tiq_instr),
                                   // instruction
                                                                                                                       19
                                                                                                                                 .u1\_SQ\_TP\_pix\_mask(u1\_SQ\_TP\_pix\_mask),
19
                                                                                                                       20
                                                                                                                                 .u2\_SQ\_TP\_pix\_mask(u2\_SQ\_TP\_pix\_mask),
                                              // TIQ FIFO pop
20
                        (tis_rtr),
                                                                                                                       21
                                                                                                                                 .u3\_SQ\_TP\_pix\_mask(u3\_SQ\_TP\_pix\_mask),
21
                                                                                                                       22
22
         .gpr_phase (gpr_phase),
                                                                                                                       23
23
                                                                                                                       24
                                                                                                                                .TP_SQ_fetch_stall(TP_SQ_fetch_stall),
                                                                                                                                                                          // stall input from TP
24
        // TIS outputs to other SQ blocks
                                                                                                                       25
      .tis_gpr_rd_addr (tis_gpr_rd_addr), // GPR read address for Fetch Address: to gpr_ra_output_mux
                                                                                                                                .busy(tis_busy),
                                           Page 61 of 92
                                                                                                                                                                   Page 62 of 92
                                                                               Ex. 2093 - sq.v
                                                                                                                                                                                                       Ex. 2093 - sq.v
         .clk(sclk_global),
                                                                                                                               wire [6:0]
                                                                                                                                                        aifl_pc_base;
 2
         .reset(srst)
                                                                                                                        2
                                                                                                                              // - aig -
                                                                                                                               wire [6:0] aiq0_pc_base;
                                                                                                                               wire [63:0] aiq0_valid_bits;
                                                                                                                               wire [101:0] aiq0 instr;
       // -- ALU Instruction Pipe 0, Pipe 1, and ALU Pipe Output Mux --
                                                                                                                              wire [2:0] aiq0_context_id;
10
                                                                                                                       10
                                                                                                                               wire [5:0] aiq0_thread_id;
11
                                                                                                                       11
12
       // - interconnect wires
                                                                                                                       12 wire [6:0] aiq1_pc_base;
13
                                                                                                                              wire [63:0] aiq1_valid_bits;
14
                                                                                                                               wire [101:0] aiq1_instr;
                                aif0_instr_rts;
      wire [0:0]
16
      wire [95:0]
                                aif0_instr;
                                                                                                                              wire [2:0] aiq1_context_id;
17
      wire ['SQ_VTX_CTL_PKT_WIDTH-1:0] aif0_ctl_pkt;
                                                                                                                              wire [5:0] aiq1_thread_id;
18
      wire [0:0]
                               aif0_thread_type;
19
       wire [5:0]
                                aif0_thread_id;
20
      wire [6:0]
                                aif0 pc base;
                                                                                                                       20 wire [0:0] ais0 acs rd rts;
21
                                                                                                                       21 wire [8:0] ais0 acs rd addr;
22
                                aif1 instr rts:
                                                                                                                       22 wire [0:0] ais0 instr start:
23
                               aif1 instr:
                                                                                                                       23
                                                                                                                              wire [0:0] ais0 instr stall:
      wire [95:0]
24
      wire ['SQ_VTX_CTL_PKT_WIDTH-1:0] aif1_ctl_pkt;
                                                                                                                       24
                                                                                                                              //wire [0:0] ais0 ld isr;
       wire [5:0]
                                aif1 thread id;
                                                                                                                       25
                                           Page 63 of 92
                                                                                                                                                                   Page 64 of 92
                                                                               Ex. 2093 - sq.v
                                                                                                                                                                                                       Ex. 2093 - sq.v
```

```
wire [0:0] ais1 acs rd rts;
 2
                                                                                                                             2
       wire [8:0] ais1 acs rd addr;
                                                                                                                                    sq_target_instr_fetch
       wire [0:0] ais1 instr start;
                                                                                                                             3
       wire [0:0] ais1_instr_stall;
                                                                                                                             4
                                                                                                                                     `SQ_VTX_CTL_PKT_WIDTH // ALU does not need lod correct bits from PIX ctl pkts
       //wire [0:0] ais1_ld_isr;
                                                                                                                             5
                                                                                                                                    u0_sq_alu_instr_fetch
 7 // this next line moved before sq_vism which now uses this signal
                                                                                                                                  .target_strap ('SQ_ALU0_STRAP), // hardwired to TEX_STRAP, ALU0_STRAP, or ALU1_STRAP
     // wire [7:0] acs_context_valid;
10
                                                                                                                            11
                                                                                                                                     // local registers
11
      wire acs_rd_rts;
                                                                                                                            12
                                                                                                                                     .inst base vtx (inst base vtx), // vertex base
12
    wire [8:0] acs_rd_addr;
                                                                                                                            13
                                                                                                                                     .inst_base_pix (inst_base_pix),
                                                                                                                                                                           // pixel base
13
       wire [2:0] acs_rd_context_id;
                                                                                                                            14
14
       wire [127:0] acs_rd_data;
                                                                                                                            15
                                                                                                                                     // cfs interface
15
       wire [0:0] aig0 rts;
                                                                                                                            16
                                                                                                                                     .cfs rts
                                                                                                                                                       (acfs0 rts),
                                                                                                                                                                            // ctl packet and ptr are valid
16
       wire [0:0] aiq1 rts;
                                                                                                                                 .cfs_ctl_pkt (acfs0_ctl_pkt),
valid_bits, gpr_base, context_id)
                                                                                                                                                                                 // the control packet (lod for pix_tex,
17
                                                                                                                                 .cfs_instr_ptr \, (acfs0_tgt_instr_ptr), // the Instruction Store address of the first target instruction
18
       wire [1:0] aif0 export info;
19
       wire [1:0] aif1 export info:
                                                                                                                            21
                                                                                                                                      .cfs\_instr\_cnt \qquad (acfs0\_tgt\_instr\_cnt), \ /\!/ \ the \ number \ of \ instructions \ to \ be \ fetched
20
       wire [1:0] aiq0 export info;
                                                                                                                                      .cfs_pc_base
                                                                                                                                                      (acfs0_state[6:0]), // the param cache base (alloc'd in arbiter)
21
       wire [1:0] aiq1_export_info;
                                                                                                                                      .cfs_thread_type (acfs0_thread_type), // vertex or pixel
22
                                                                                                                                      .cfs_thread_id (acfs0_status[21:16]), //
23
                                                                                                                            25
                                                                                                                                     //.cfs_pulse_sx (acfs0_status[13]), //
24
       // -- ALU Instruction Fetch 0 --
                                                                                                                                  //.cfs_export_id (acfs0_state[7]), // export_id that cfs0 is pushing down pipe 0 (sets global export_id)
25
                                              Page 65 of 92
                                                                                                                                                                          Page 66 of 92
                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                               Ex. 2093 - sq.v
          .cfs_export_info ({acfs0_state[7], acfs0_status[13]}), // export_info = {exp_id, pulse_sx}
                                                                                                                                      .clk(sclk_global),
          .cfs_last_in_thread(acfs0_status[12]), // last instr in shader prog
 2
                                                                                                                             2
                                                                                                                                      .reset(srst)
                          (aif0 rtr).
                                         // AIF0 can take a new packet
                                                                                                                             3
          .tif rtr
 4
         // instruction store interface
          is read addr (aif0 is read addr), // instruction store read address
                                                                                                                                    // -- ALU Instruction Oueue 0 --
          .is_read_data (is_read_data),// instruction store read data
          .is_phase (is_phase), // instruction store phase
          .alu_phase (alu_phase), // tied low for TEX instance, to alu_phase for ALU
10
                                                                                                                            10 parameter AIQ_NUM_WORDS = 4;
11
                                                                                                                            11
                                                                                                                                    parameter AIQ_ADDR_BITS = 2;
12
         // outputs to the target instruction decoder/queue
                                                                                                                            12
13
          .tif_pc_base_q (aif0_pc_base), // the target control packet (pipelined from input)
14
          .tif_ctl_pkt_q (aif0_ctl_pkt), // the target control packet (pipelined from input)
                                                                                                                                   // #(AIQ_NUM_WORDS, AIQ_ADDR_BITS)
15
         .tif_export_info_q(aif0_export_info), //
                                                                                                                            15
                                                                                                                                    u0_sq_alu_instr_queue
16
         .tif\_last\_in\_thread\_q(aif0\_last\_in\_thread), \\ \hspace*{0.2in} \textit{// last instruction flag}
17
          .tif_last_in_group_q(aif0_last_in_group), // last instruction flag
                                                                                                                                     .write_rts (aif0_instr_rts),
18
          .tif_thread_type_q(aif0_thread_type), // 0: pixel, 1: vertex
                                                                                                                            18
                                                                                                                                     .write rtr (aiq0 rtr),
          .tif_thread_id_q (aif0_thread_id),
                                                    // the target thread id (pipelined from
19
20
      input)
                                                                                                                            20
                                                                                                                                     // AIQ inputs
21
         .tif_instr_q(aif0_instr),
                                                // the target instruction data register (TIR)
                                                                                                                                     //write data(),
                                                                                                                            21
22
         .tif\_instr\_rts\_q \qquad (aif0\_instr\_rts),
                                                  // the target instr register is valid
                                                                                                                            22
      .tiq_rtr (aiq0_rtr),
(and other pipeline data)
                                                                                                                                     .aif export info (aif0 export info),
                                                // the target instr queue is ready to take the AIR
23
24
                                                                                                                            23
                                                                                                                                      .aif_pc_base_q (aif0_pc_base),
                                                                                                                                                                                   // param cache base
25
                                                                                                                            24
                                                                                                                                      .aif\_last\_in\_thread\_q(aif0\_last\_in\_thread), \qquad \textit{// last instruction flag}
          .busy(aif0_busy),
                                                                                                                                      .aif\_last\_in\_group\_q(aif0\_last\_in\_group),
                                                                                                                                                                                   // last instruction flag
                                             Page 67 of 92
                                                                                                                                                                          Page 68 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                               Ex. 2093 - sq.v
```

```
.aif_thread_type_q(aif0_thread_type), \hspace{0.2cm} /\!\!/ \hspace{0.1cm} 0: pixel, 1: vertex
 2
                                            // control packet
          .aif_ctl_pkt_q (aif0_ctl_pkt),
          .aif\_thread\_id\_q \quad (aif0\_thread\_id), \qquad \textit{// thread\_id}
                                                                                                                                   // -- ALU Instruction Sequencer 0 --
          .aif_instr_q (aif0_instr), // instruction register (TIR)
          .read rts
                          (aiq0_rts),
                                                                                                                                    sq_alu_instr_seq
                                                                                                                                    u0_sq_alu_instr_seq
                          // {control packet, clause num, instruction}
                                                                                                                                     .alu_strap (LO),
10
                                                                                                                            10
11
          .aiq_export_info (aiq0_export_info), // {exp_id, pulse_sx}
                                                                                                                            11
                                                                                                                                     // AIQ interface
12
          .aiq_pc_base (aiq0_pc_base), // param cache base addr
                                                                                                                                     .aiq_rts (aiq0_rts),
                                                                                                                                                                           // rts from AIQ FIFO
13
          .aiq_last_in_thread(aiq0_last_in_thread),// last instruction flag
                                                                                                                                     .aiq_export_info (aiq0_export_info), // {exp_id, pulse_sx}
14
          .aiq_last_in_group(aiq0_last_in_group), // last instruction flag
                                                                                                                                      .aiq_last_in_thread(aiq0_last_in_thread),// last instruction flag
15
                                                                                                                            15
                                                                                                                                     .aiq last in group(aiq0 last in group), // last instruction flag
                                                      // context_id (from ctl packet)
16
          .aiq context id (aiq0 context id),
                                                                                                                            16
                                                                                                                                      .aiq context id (aiq0 context id),
                                                                                                                                                                                  // context_id (from ctl packet)
17
          .aiq_valid_bits (aiq0_valid_bits), // valid bits (from ctl packet)
                                                                                                                            17
                                                                                                                                      .aiq_thread_id (aiq0_thread_id),
                                                                                                                                                                                 // thread id
                                                                                                                                      .aiq_thread_type (aiq0_thread_type), // thread_id
18
         .aiq_thread_id (aiq0_thread_id),
                                                    // thread_id
                                                                                                                            18
                                                 // thread_id
19
          .aiq_thread_type (aiq0_thread_type),
                                                                                                                            19
                                                                                                                                     .aiq_instr (aiq0_instr), // instruction
20
          .aiq_instr (aiq0_instr),
                                    // instruction
                                                                                                                            20
                                                                                                                                                         // AIQ FIFO pop
21
                                                                                                                            21
                                                                                                                                     .ais rtr(ais0 rtr),
22
          .clk(sclk_global),
23
         .reset(srst)
                                                                                                                            23
                                                                                                                                     // phase inputs
24
                                                                                                                            24
                                                                                                                                      .gpr_phase (gpr_phase),
25
                                                                                                                            25
                                                                                                                                     .alu_phase (alu_phase),
                                              Page 69 of 92
                                                                                                                                                                         Page 70 of 92
                                                                                   Ex. 2093 - sq.v
                                                                                                                                                                                                              Ex. 2093 - sq.v
 2
         // AIS outputs to shader seq
          .ais done (ais0 done),
                                                                                                                                   sq target instr fetch
          .ais thread type q(ais0 thread type),
                                                                                                                             4
          .ais_thread_id_q (ais0_thread_id),
                                                                                                                             5
                                                                                                                                     'SQ VTX CTL PKT WIDTH
                                                                                                                             6
                                                                                                                             7
          .ais free done
                                (ais0 free done),
                                                                                                                                    ul_sq_alu_instr_fetch
          .ais_free_id_q
                                 (ais0 free id),
                                                                                                                             8
                                                                                                                                .target_strap (`SQ_ALU1_STRAP), // hardwired to TEX_STRAP, ALU0_STRAP, or ALU1_STRAP
10
         // to AIS output module
11
          .ais_instr_start (ais0_instr_start),
                                                                                                                                     // local registers
12
          .ais_instr_stall (ais0_instr_stall),
                                                                                                                                    .inst_base_vtx (inst_base_vtx),
13
          .ais_acs_rd_rts (ais0_acs_rd_rts),
                                                                                                                                     .inst_base_pix (inst_base_pix), // pixel base (only in separate mode)
14
          .ais_acs_rd_addr (ais0_acs_rd_addr),
15
                                                                                                                                     // cfs interface
16
         //.aluconst_context_valid(acs_context_valid),
                                                                                                                            17
                                                                                                                                     .cfs_rts (acfs1_rts),
                                                                                                                                                                            // ctl packet and ptr are valid
17
                                                                                                                                                     (acfs1_ctl_pkt),
                                                                                                                                                                              // the control packet (lod for pix_tex,
                                                                                                                                      .cfs ctl pkt
18
         .busy(ais0_busy),
                                                                                                                                 valid_bits, gpr_base, context_id)
19
         .clk(sclk_global),
                                                                                                                                     .cfs\_instr\_ptr \qquad (acfs1\_tgt\_instr\_ptr), \ /\!/ \ the \ Instruction \ Store \ address \ of \ the \ first \ target
20
         .reset(srst)
                                                                                                                                     .cfs_instr_cnt (acfs1_tgt_instr_cnt), // the number of instructions to be fetched
21
                                                                                                                            23
                                                                                                                                                      (acfs1_state[6:0]), // the param cache base (alloc'd in arbiter)
22
                                                                                                                            24
                                                                                                                                     //.cfs_pulse_sx (acfs1_status[13]), //
23
                                                                                                                                 \label{eq:continuous} \mbox{$\#$/$cfs_export_id} \quad (acfs1\_state[7]), \qquad \mbox{$\#$/$export_id$ that cfs0 is pushing down pipe 0 (sets global export_id)}
24
       // -- ALU Instruction Fetch 1 --
                                                                                                                                     .cfs\_export\_info \quad (\{acfs1\_state[7], acfs1\_status[13]\}), \ // \ export\_info = \{exp\_id, pulse\_sx\}
                                             Page 71 of 92
                                                                                                                                                                         Page 72 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                              Ex. 2093 - sq.v
```

```
.cfs thread type (acfs1 thread type), // vertex or pixel
                                                                                                                                     .clk(sclk global),
2
         .cfs thread id (acfs1 status[21:16]), //
                                                                                                                                     .reset(srst)
         .cfs last in thread(acfs1 status[12]), // last instr in shader prog
                         (aif1 rtr), // AIF1 can take a new packet
         // instruction store interface
         .is_read_addr (aif1_is_read_addr), // instruction store read address
                                                                                                                                   // -- ALU Instruction Queue 1 --
         .is_read_data (is_read_data),// instruction store read data
         .is_phase (is_phase), // instruction store phase
         .alu_phase (alu_phase), // tied low for TEX instance, to alu_phase for ALU instances
                                                                                                                                  sq_alu_instr_queue
11
                                                                                                                           11
                                                                                                                                   //#(AIQ_NUM_WORDS, AIQ_ADDR_BITS)
12
         // outputs to the target instruction decoder/queue
                                                                                                                                  u1_sq_alu_instr_queue
13
         .tif_pc_base_q (aifl_pc_base), // the target control packet (pipelined from input)
                                                                                                                           13
14
         .tif_ctl_pkt_q (aifl_ctl_pkt), // the target control packet (pipelined from input)
                                                                                                                                    .write_rts (aif1_instr_rts),
15
         .tif export info q(aif1 export info), //
                                                                                                                           15
                                                                                                                                    write rtr (aigl rtr),
         .tif\_last\_in\_group\_q(aifl\_last\_in\_group), \\ \hspace*{0.2in} /\!/\ last\ instruction\ flag
16
                                                                                                                           16
17
         .tif\_last\_in\_thread\_q(aifl\_last\_in\_thread), \\ \hspace*{0.2in} \textit{// last instruction flag}
                                                                                                                           17
                                                                                                                                    // AIO inputs
18
         .tif_thread_type_q(aifl_thread_type), // 0: pixel, 1: vertex
                                                                                                                           18
                                                                                                                                    //write data(),
19
20
         .tif_thread_id_q (aif1_thread_id),
                                               // the target thread_id (pipelined from
                                                                                                                           19
                                                                                                                                     .aif_export_info (aifl_export_info),
                                                                                                                                                                                 // control packet
                                                                                                                                                                                // control packet
                                                                                                                           20
                                                                                                                                    .aif_pc_base_q (aifl_pc_base),
21
         .tif_instr_q(aif1_instr),
                                            // the target instruction data register (TIR)
                                                                                                                                     .aif_last_in_thread_q(aifl_last_in_thread), // last instruction flag
                                                                                                                           21
22
         .tif_instr_rts_q (aif1_instr_rts),
                                                   // the target instr register is valid
                                                                                                                           22
                                                                                                                                    .aif_last_in_group_q(aifl_last_in_group), // last instruction flag
     .tiq_rtr (aiq1_rtr),
(and other pipeline data)
                                               // the target instr queue is ready to take the AIR
                                                                                                                           23
                                                                                                                                    .aif_ctl_pkt_q (aifl_ctl_pkt), // control packet
25
                                                                                                                           24
                                                                                                                                     .aif_thread_type_q(aif1_thread_type), // 0: pixel, 1: vertex
         .busy(aif1_busy),
                                                                                                                            25
                                                                                                                                     .aif_thread_id_q (aifl_thread_id), // thread_id
                                             Page 73 of 92
                                                                                                                                                                         Page 74 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                              Ex. 2093 - sq.v
         .aif_instr_q
                         (aifl_instr),
                                               // instruction register (TIR)
2
                                                                                                                             2
                                                                                                                                   sq alu instr seq
         read rts
                          (aiq1 rts),
                                                                                                                                   ul_sq_alu_instr_seq
 4
                          (ais1 rtr),
                                                                                                                             4
         .read rtr
         //read data(),
                             // {control packet, clause num, instruction}
                                                                                                                                     .alu strap (HI),
                                                                                                                                                                   // whether ALU 0 or ALU 1
        // AIQ outputs
                                                                                                                                     // AIO interface
 8
         .aiq\_export\_info \quad (aiq1\_export\_info), \quad \textit{//} \{exp\_id, pulse\_sx\}
                                                                                                                                     .aiq rts (aiq1 rts),
                                                                                                                                                                           // rts from AIO FIFO
         .aiq_pc_base (aiq1_pc_base), // param cache base addr
                                                                                                                                     .aiq\_export\_info \quad (aiq1\_export\_info), \quad \textit{//} \{exp\_id, pulse\_sx\}
         .aiq\_last\_in\_thread(aiq1\_last\_in\_thread), //\ last\ instruction\ flag
10
                                                                                                                            10
                                                                                                                                     .aiq_last_in_thread(aiq1_last_in_thread),// last instruction flag
11
         .aiq\_last\_in\_group(aiq1\_last\_in\_group), \ /\!/\ last\ instruction\ flag
                                                                                                                            11
                                                                                                                                     .aiq\_last\_in\_group(aiq1\_last\_in\_group), \ //\ last\ instruction\ flag
12
         .aiq_context_id (aiq1_context_id), // context_id (from ctl packet)
                                                                                                                            12
                                                                                                                                     .aiq_context_id (aiq1_context_id), // context_id (from ctl packet)
                                                                                                                                      . aiq\_thread\_id \qquad (aiq1\_thread\_id), \qquad \textit{// thread\_id}
13
         .aiq_valid_bits (aiq1_valid_bits), // valid bits (from ctl packet)
         .aiq_thread_type (aiq1_thread_type), // 0: pixel, 1: vertex
                                                                                                                                      .aiq_thread_type (aiq1_thread_type), // thread type (vtx/pix)
15
         .aiq_thread_id (aiq1_thread_id), // thread_id
                                                                                                                                      .aiq_instr (aiq1_instr), // instruction
16
         .aiq_instr (aiq1_instr),
                                   // instruction
17
                                                                                                                                                (aisl_rtr),
                                                                                                                                                                        // AIQ FIFO pop
18
         .clk(sclk_global),
                                                                                                                            18
19
         .reset(srst)
                                                                                                                                     // phase inputs
20
                                                                                                                            20
                                                                                                                                                                    // GPR phase
                                                                                                                                     .gpr phase (gpr phase),
21
                                                                                                                            21
                                                                                                                                     .alu phase (alu phase),
22
                                                                                                                            22
23
                                                                                                                            23
                                                                                                                                     // AIS outputs to shader seq
                                                                                                                            24
                                                                                                                                     .ais_done (ais1_done), //
24
       // -- ALU Instruction Sequencer 1 --
                                                                                                                            25
                                                                                                                                     .ais\_thread\_type\_q(ais1\_thread\_type), \quad //
                                             Page 75 of 92
                                                                                                                                                                         Page 76 of 92
                                                                                  Ex. 2093 - sq.v
                                                                                                                                                                                                              Ex. 2093 - sq.v
```

```
ais thread id q (ais1 thread id), //
                                                                                                                           1 (
 2
                                                                                                                                  // AIQ inputs
          .ais free done (ais1 free done),
                                                                                                                                  .aiq0_export_id (aiq0_export_info[1]),//
         .ais\_free\_id\_q \qquad (ais \, l\_free\_id),
                                                                                                                                  .aiq0_pc_base (aiq0_pc_base),
                                                                                                                                                                               // param cache base ptr
                                                                                                                                   .aiq0_valid_bits (aiq0_valid_bits), // valid bits (from ctl packet)
         // to AIS output module
                                                                                                                                   .aiq0_context_id (aiq0_context_id), // context_id (from ctl packet)
         .ais_instr_start (ais1_instr_start),
                                                                                                                                   .aiq0_instr (aiq0_instr), // instruction
          .ais_instr_stall (ais1_instr_stall),
                                                                                                                                   .aiq0_gpr_rd_en (aiq0_rts),
          .ais_acs_rd_rts (ais1_acs_rd_rts),
                                                                                                                                  .aiq1_export_id (aiq1_export_info[1]),//
         .ais_acs_rd_addr (ais1_acs_rd_addr),
11
                                                                                                                                  .aiq1_pc_base (aiq1_pc_base),
12
         //.aluconst_context_valid(acs_context_valid),
                                                                                                                                  .aiq1_valid_bits (aiq1_valid_bits),
13
                                                                                                                                  .aiql_context_id (aiql_context_id),
14
         .busy(ais1_busy),
                                                                                                                                  .aiq1_instr (aiq1_instr),
15
         .clk(sclk global),
                                                                                                                         15
                                                                                                                                  .aiql gpr rd en (aiql rts),
16
         .reset(srst)
                                                                                                                         16
17
                                                                                                                         17
                                                                                                                                  // AIS inputs
18
                                                                                                                         18
                                                                                                                                  .ais0_acs_rd_rts (ais0_acs_rd_rts), // alu const store read addr valid
19
                                                                                                                         19
                                                                                                                                  .ais0 acs rd addr (ais0 acs rd addr), // alu constant store read address (from instr)
20
                                                                                                                         20
                                                                                                                                  .ais0_instr_start (ais0_instr_start), // just OR these guys before reg to SP
21
      // -- ALU Instruction Sequencer Output Mux --
                                                                                                                         21
                                                                                                                                  .ais0 instr stall (ais0 instr stall),
22
                                                                                                                         22
                                                                                                                                  //.ais0 ld isr(LO),
23
                                                                                                                         23
24
       sq_ais_output
                                                                                                                         24
                                                                                                                                  .aisl_acs_rd_rts (aisl_acs_rd_rts),
       u_sq_ais_output
                                                                                                                                  .ais1_acs_rd_addr (ais1_acs_rd_addr),
                                            Page 77 of 92
                                                                                                                                                                      Page 78 of 92
                                                                                 Ex. 2093 - sq.v
                                                                                                                                                                                                          Ex. 2093 - sq.v
                                                                                                                             .SQ_SP_gpr_wr_en ({u0_SQ_SP_gpr_vr_u2_SQ_SP_gpr_wr_en, u3_SQ_SP_gpr_wr_en}),
          .ais1_instr_start (ais1_instr_start),
                                                                                                                                                           ({u0_SQ_SP_gpr_wr_en,
                                                                                                                                                                                                 u1_SQ_SP_gpr_wr_en,
 2
         .ais1 instr stall (ais1 instr stall),
                                                                                                                                  . SQ\_SP\_gpr\_rd\_addr (SQ\_SP\_gpr\_rd\_addr),
         //.ais1_ld_isr(LO),
                                                                                                                                  .SQ\_SP\_gpr\_rd\_en \qquad \quad (SQ\_SP\_gpr\_rd\_en),
                                                                                                                                  . SQ\_SP\_gpr\_phase \qquad \quad (SQ\_SP\_gpr\_phase\_mux),
         // other inputs that get muxed out to the SP or SX
                                                                                                                                   . SQ\_SP\_gpr\_input\_sel(SQ\_SP\_gpr\_input\_mux),
         .tis_gpr_rd_addr (tis_gpr_rd_addr), // texture fetch read address
                                                                                                                                   . SQ\_SP\_gpr\_channel\_mask (SQ\_SP\_channel\_mask),
          .tis_gpr_rd_en (tiq_rts), // texture fetch read enable
      .ia_vertex_sel (ia_vertex_sel), // select VISM gpr write address vs. PISM gpr write address
                                                                                                                                   . SQ\_SP\_instr\_start(SQ\_SP\_instruct\_start), \\
10
          .vi_gpr_wr_addr (vi_gpr_wr_addr), // VISM gpr write address
                                                                                                                         10
                                                                                                                                   .SQ_SP_instr_stall(SQ_SP_stall),
                                                 // VISM gpr write enable
11
         .vi_gpr_wr_en (vi_gpr_wr_en),
                                                                                                                                   .SQ_SP_instr (SQ_SP_instruct),
                                                                                                                                   .SQ_SP_const (SQ_SP_const),
12
         .pi_gpr_wr_addr (pi_gpr_wr_addr), // PISM gpr write address
13
                                                                                                                         13
         .pi_gpr_wr_en (pi_gpr_wr_en),
                                                 // PISM gpr write enable
14
                                                                                                                         14
15
         // phase inputs
                                                                                                                         15
                                                                                                                                  .SQ_SP_exporting(SQ_SP_exporting),
16
         .gpr_phase(gpr_phase),
                                                                                                                                  .SQ_SP_exp_id (SQ_SP_exp_id),
17
                                                                                                                         17
                                                                                                                                   .u0_SQ_SP_write_mask(u0_SQ_SP_pix_mask),
         .alu phase(alu phase),
18
                                                                                                                                  .u1_SQ_SP_write_mask(u1_SQ_SP_pix_mask),
19
         // ALU Const Store Interface
                                                                                                                         19
                                                                                                                                   .u2 SQ SP write mask(u2 SQ SP pix mask),
20
                                                                                                                         20
                                                                                                                                   .u3\_SQ\_SP\_write\_mask(u3\_SQ\_SP\_pix\_mask),
          .acs rd rts(acs rd rts),
                                       // alu constant store read address valid
21
          .acs rd addr (acs rd addr), // alu constant store read address
                                                                                                                         21
22
          .acs_rd_context_id (acs_rd_context_id), // alu constant store read state (context)
                                                                                                                         22
                                                                                                                                  // outputs to SX
23
         .acs_rd_data (acs_rd_data), // alu constant store read data
                                                                                                                         23
                                                                                                                                  .SQ_SX_pc_wr_addr (SQ_SX_pc_wr_addr),
24
                                                                                                                         24
                                                                                                                                   . SQ\_SX\_pc\_wr\_en \qquad \quad (SQ\_SX\_pc\_wr\_en),
25
         // outputs to SP
                                                                                                                         25
                                                                                                                                   . SQ\_SX\_pc\_channel\_mask (SQ\_SX\_pc\_channel\_mask),
          . SQ\_SP\_gpr\_wr\_addr (SQ\_SP\_gpr\_wr\_addr),
                                                                                                                         26
                                           Page 79 of 92
                                                                                                                                                                      Page 80 of 92
                                                                                Ex. 2093 - sq.v
                                                                                                                                                                                                          Ex. 2093 - sq.v
```

```
.clk(sclk global),
  2
                 .reset(srst)
                                                                                                                                                                                                                                sq_rbbm_interface
                                                                                                                                                                                                                                 u\_sq\_rbbm\_interface
                                                                                                                                                                                                                     4
                                                                                                                                                                                                                                   // RBBM
                                                                                                                                                                                                                                    .RBBM_a
                                                                                                                                                                                                                                                                 (RBBM_a),
                                                                                                                                                                                                                                    .RBBM_wd
                                                                                                                                                                                                                                                                        (RBBM_wd),
                                                                                                                                                                                                                                    .RBBM_we
                                                                                                                                                                                                                                                                            (RBBM_we),
             // -- RBBM Interface, Local Registers --
                                                                                                                                                                                                                                    .RBBM_be
                                                                                                                                                                                                                                                                            (RBBM_be),
                                                                                                                                                                                                                                    .RBBM_re
                                                                                                                                                                                                                                                                (RBBM_re),
11
                                                                                                                                                                                                                                    .RBBM_SQ_soft_reset(RBBM_SQ_soft_reset),
12
                                                                                                                                                                                                                                    . SQ\_RBBM\_nrtrtr(SQ\_RBBM\_nrtrtr),
13
             wire [31:0] rbi_data;
                                                                                                                                                                                                                    13
                                                                                                                                                                                                                                    .SQ_RBBM_rtr (SQ_RBBM_rtr),
14
             wire [14:0] rbi_addr;
                                                                                                                                                                                                                                    .SQ_RBBM_cntx0_busy (SQ_RBBM_cntx0_busy),
15
                                                                                                                                                                                                                   15
                                                                                                                                                                                                                                    .SQ_RBBM_cntx17_busy (SQ_RBBM_cntx17_busy),
             wire [00:0] rbi is rts;
16
             wire [00:0] rbi is rtr;
                                                                                                                                                                                                                   16
17
                                                                                                                                                                                                                   17
                                                                                                                                                                                                                                   //RBBM read data daisy chain
             wire [00:0] rbi tcs rts;
18
             wire [00:0] rbi_tcs_rtr;
                                                                                                                                                                                                                   18
                                                                                                                                                                                                                                    .RBB rs in
                                                                                                                                                                                                                                                               (RBB rs)
19
             wire [00:0] rbi acs rts;
                                                                                                                                                                                                                   19
                                                                                                                                                                                                                                    .RBB rs out
                                                                                                                                                                                                                                                                (SQ RBB rs).
20
             wire [00:0] rbi acs rtr;
                                                                                                                                                                                                                   20
                                                                                                                                                                                                                                    .RBB rd in
                                                                                                                                                                                                                                                                 (RBB rd).
21
                                                                                                                                                                                                                                    .RBB_rd_out (SQ_RBB_rd),
             wire [00:0] rbi_draw_command;
                                                                                                                                                                                                                   21
22
             wire~[15:0]~sq\_busy\_bits = \{vtx\_write\_busy, pix\_write\_busy, vtx\_tb\_busy, pix\_write\_busy, pix_write\_busy, pix_write\_busy, pix_write\_busy, pix_write\_busy, pix_write\_busy, pix_write\_busy, pix_write_busy, pix
23
                                                     pix_tb_busy, tcfs_busy, acfs0_busy, acfs1_busy,
                                                                                                                                                                                                                   23
                                                                                                                                                                                                                                   // common
24
                                                     tif_busy, tis_busy, aif0_busy, aif1_busy,
                                                                                                                                                                                                                   24
                                                                                                                                                                                                                                    .o_rbi_data
                                                                                                                                                                                                                                                                 (rbi_data),
25
                                                     ais0\_busy, ais1\_busy, aiq0\_rts, aiq1\_rts, tiq\_rts\};
                                                                                                                                                                                                                                    .o_rbi_addr
                                                                                                                                                                                                                                                                 (rbi_addr),
                                                                             Page 81 of 92
                                                                                                                                                                                                                                                                                                 Page 82 of 92
                                                                                                                                            Ex. 2093 - sq.v
                                                                                                                                                                                                                                                                                                                                                                Ex. 2093 - sq.v
                 .o context switch (rbi draw command),
                                                                                                                                                                                                                                    .o inst base vtx
                                                                                                                                                                                                                                                                            (inst base vtx),
 2
                                                                                                                                                                                                                                                                            (inst base pix),
                .i_sq_busy_bits (sq_busy_bits),
                                                                                                                                                                                                                                    .o inst base pix
                // aluconst
                                                                                                                                                                                                                                    .o\_sq\_wrapping\_l\_set \hspace{0.5cm} (sq\_wrapping\_l\_set),
                 .o aluconst rts (rbi acs rts),
                                                                                                                                                                                                                                    .o_sq_wrapping_0_set (sq_wrapping_0_set),
                 .i_aluconst_rtr (rbi_acs_rtr),
                                                                                                                                                                                                                                   //.o_sampling_mode_set (sampling_mode_set),
                                                                                                                                                                                                                                   //.o_gen_index_set (gen_index_set),
                // texconst
                                                                                                                                                                                                                                   //.o_param_gen_i0_set (param_gen_i0_set),
                 .o_texconst_rts (rbi_tcs_rts),
                                                                                                                                                                                                                                   //.o_num_param_set (vs_exp_count7_set),
                 .i_texconst_rtr (rbi_tcs_rtr),
 10
                                                                                                                                                                                                                   10
                                                                                                                                                                                                                                    .i_clk
                                                                                                                                                                                                                                                                (sclk_global),
 11
                // instr store
                                                                                                                                                                                                                   11
12
                 .o_ins_rts (rbi_is_rts),
                                                                                                                                                                                                                   12
13
                 .i_ins_rtr (rbi_is_rtr),
                                                                                                                                                                                                                   13
                                                                                                                                                                                                                                // New State Register Outputs
 14
 15
                                                                                                                                                                                                                               .o_vs_program_base_set
                                                                                                                                                                                                                                                                                        (vs_program_base_set),
 16
                //.o_cfc_rts (),
                                                                                                                                                                                                                                                                                      (),
                                                                                                                                                                                                                               .o_vs_program_size_set
17
                .i_cfc_rtr (LO),
                                                                                                                                                                                                                              .o_ps_program_base_set
                                                                                                                                                                                                                                                                                        (ps_program_base_set),
 18
                                                                                                                                                                                                                               .o ps program size set
 19
                // local registers
                                                                                                                                                                                                                                .o sq cf program size vs cf size set (),
20
                                                                                                                                                                                                                   20
               //.ps num reg set
                                                          (ps num reg set),
                                                                                                                                                                                                                                .o_sq_cf_program_size_ps_cf_size_set (),
21
               //.vs num reg set
                                                                                                                                                                                                                   21
                                                                                                                                                                                                                                .o interpolator cntl param shade set (param shade set),
                                                         (vs num reg set),
22
                                                                                                                                                                                                                   22
                //.param shade set
                                                                     (param shade set),
                                                                                                                                                                                                                                .o\_interpolator\_cntl\_sampling\_pattern\_set(),\\
23
                                                                                                                                                                                                                   23
                                                                                                                                                                                                                                .o\_program\_cntl\_vs\_num\_reg\_set \ (vs\_num\_reg\_set),
24
                                                                                                                                                                                                                   24
               //.o_ps_base_set
                                                          (ps base set),
                                                                                                                                                                                                                                .o_program_cntl_ps_num_reg_set (ps_num_reg_set),
25
               //.o_vs_base_set
                                                          (vs_base_set),
                                                                                                                                                                                                                                 .o_program_cntl_vs_resource_set (vs_resource_set),
                                                                             Page 83 of 92
                                                                                                                                                                                                                                                                                                 Page 84 of 92
                                                                                                                                            Ex. 2093 - sq.y
                                                                                                                                                                                                                                                                                                                                                                Ex. 2093 - sq.v
```

```
.o_program_cntl_ps_resource_set (ps_resource_set),
                                                                                                                       .o_ps_const_size_set
2
      .o_program_cntl_param_gen_i0_set (param_gen_i0_set),

    o_context_misc_inst_pred_optimize_set(),

      .o_program_cntl_gen_index_set (gen_index_set),
                                                                                                                        .o_cf_rd_base_rd_base_set (),
       .o_program_cntl_vs_export_count_set (vs_export_count_set),
                                                                                                                        .o_provoking_vtx_provoking_vtx_set (),
       .o\_program\_cntl\_vs\_export\_mode\_set \qquad (vs\_export\_mode\_set),
                                                                                                                        .o_debug_misc_0_db_prob_on_set (),
       .o\_program\_cntl\_ps\_export\_mode\_set \qquad (ps\_export\_mode\_set),
                                                                                                                        .o_debug_misc_0_db_prob_break_set (),
                                                                                                                       .o_debug_misc_0_db_prob_addr_set (),
       .o_wrapping_0_param_wrap_0_set(),
                                                                                                                       .o_debug_misc_0_db_prob_count_set (),
       .o_wrapping_0_param_wrap_1_set(),
      .o_wrapping_0_param_wrap_2_set(),
                                                                                                                       .o_debug_misc_l_db_on_pix_set (),
      .o_wrapping_0_param_wrap_3_set(),
                                                                                                                        .o_debug_misc_l_db_on_vtx_set (),
11
                                                                                                                       .o_debug_misc_1_db_inst_count_set (),
      .o_wrapping_0_param_wrap_4_set(),
12
      .o_wrapping_0_param_wrap_5_set(),
                                                                                                                       .o_debug_misc_l_db_break_addr_set ()
13
      .o wrapping 0 param wrap 6 set(),
14
      .o wrapping 0 param wrap 7 set(),
15
                                                                                                                 15
      .o wrapping 1 param wrap 8 set(),
16
                                                                                                                 16
       .o_wrapping_l_param_wrap_9_set(),
17
                                                                                                                 17
                                                                                                                        // -- Instruction Store, ALU and Texture Constant Stores --
       .o_wrapping_l_param_wrap_10_set (),
18
       .o_wrapping_1_param_wrap_11_set
                                                                                                                 18
19
       .o_wrapping_1_param_wrap_12_set
                                           (),
                                                                                                                 19
20
       .o_wrapping_1_param_wrap_13_set
                                           (),
                                                                                                                 20
21
       .o_wrapping_1_param_wrap_14_set
                                           (),
                                                                                                                 21
                                                                                                                        wire [01:0] texconst_phase;
22
       .o_wrapping_1_param_wrap_15_set
                                           (),
                                                                                                                 22
23
       .o_vs_const_base_set
                                           0.
                                                                                                                 23
24
       .o_vs_const_size_set
                                                                                                                 24
                                                                                                                        // -- Instruction Store --
       .o_ps_const_base_set
                                           (),
                                         Page 85 of 92
                                                                                                                                                           Page 86 of 92
                                                                           Ex. 2093 - sq.v
                                                                                                                                                                                             Ex. 2093 - sq.v
                                                                                                                          .i reset
                                                                                                                                          (srst)
2
                                                                                                                  2
      sq instruction store
3
       u_sq_inst_store
4
5
        // memory access phase control
                                                                                                                        // -- Texture Constant Store --
        .i_is_phase(is_phase),
7
        .i_is_sub_phase (is_subphase),
        // RBI
                                                                                                                        sq_texconst_top
         .i_rbi_data (rbi_data),
10
                                                                                                                 10
11
         .i_rbi_addr(rbi_addr[14:0]),
                                                                                                                 11
12
         .i_rts (rbi_is_rts),
                                                                                                                 12
                                                                                                                         // from RBI
13
                       (rbi_is_rtr),
                                                                                                                          .i_data_in (rbi_data),
         .o_rtr
14
                                                                                                                          .i_addr_in (rbi_addr[7:3]),
15
                                                                                                                          .i_rts (rbi_tcs_rts),
16
        .i_tex_cf_addr (tcfs_is_read_addr),
                                                                                                                                       (rbi_tcs_rtr),
                                                                                                                          .o_rtr
17
        .i_alu0_cf_addr (acfs0_is_read_addr),
18
         .i_alu1_cf_addr (acfs1_is_read_addr),
                                                                                                                 18
                                                                                                                          .i context switch temp(rbi draw command),
19
         .i_tex_addr (tif_is_read_addr),
20
        .i alu0 addr (aif0 is read addr),
                                                                                                                 20
                                                                                                                          // From SQ
                                                                                                                     .i_sq_read_laddr (texconst_rd_addr), \ensuremath{//} this really comes directly from TIQ (now passes thru TIS)
21
        .i_alu1_addr (aif1_is_read_addr),
22
                                                                                                                 23
                                                                                                                          . i\_sq\_read\_context(tiq\_context\_id),
23
         .o is data (is read data),
                                                                                                                 24
                                                                                                                          .i_texconst_phase (texconst_phase),
                                                                                                                                                            // two cycles for read (0,1) and two for write (2,3)
24
                                                                                                                 25
                                                                                                                          .o_read_data (texconst_rd_data),
25
         .i clk
                        (sclk_global),
                                                                                                                 26
                                         Page 87 of 92
                                                                                                                                                           Page 88 of 92
                                                                           Ex. 2093 - sq.y
                                                                                                                                                                                            Ex. 2093 - sq.v
```

```
//.o_context_valid (),
                                                                                                                // AIS output read interface
 1
                                                                                                             .i_sq_read_laddr (acs_rd_addr), /\!/ this really comes directly from TIQ (now passes thru TIS)
2
3
       .i_sq_context_done(pix_state_change),
                                                                                                                 .i_sq_read_context(acs_rd_context_id),
      .i_sq_context (pix_old_state),
                                                                                                                 .o_read_data (acs_rd_data),
      .i_clk
                (sclk_global),
                                                                                                                .o_context_valid (acs_context_valid),
        .i_reset
                                                                                                                .i_sq_context_done(pix_state_change),
                                                                                                                .i_sq_context (pix_old_state),
10
                                                                                                         11
11
                                                                                                         12
                                                                                                                .i_clk
                                                                                                                            (sclk_global),
12 // -- ALU Constant Store --
                                                                                                                .i_reset (srst)
                                                                                                         13
13 //-----
                                                                                                         14
14
                                                                                                         15
15
     sq_aluconst_top
                                                                                                         16
16
      u_sq_aluconst_top
                                                                                                         17
17
                                                                                                         18
18
      // RBI interface
                                                                                                         19
                                                                                                               // -- Miscellaneous --
19
       .i data in (rbi data),
                                                                                                         20
20
        .i_addr_in (rbi_addr[10:4]),
                                                                                                         21
21
      .i_rts (rbi_acs_rts),
                                                                                                         22
22
                      (rbi_acs_rtr),
                                                                                                         23 // -----
23
                                                                                                         24
                                                                                                               // -- Phase Generation --
24
        .i_context_switch_temp(rbi_draw_command),
                                                                                                         25
25
                                      Page 89 of 92
                                                                                                                                               Page 90 of 92
                                                                      Ex. 2093 - sq.v
                                                                                                                                                                              Ex. 2093 - sq.v
     sq phase gen
2
     u_sq_phase_gen
3
4
        .gpr_phase (gpr_phase),
       .texconst_phase (texconst_phase),
       .is_phase (is_phase),
        .is_subphase (is_subphase),
        .alu_phase (alu_phase),
        .cfs_phase (cfs_phase),
        .state_read_phase (state_read_phase),
10
11
12
      .clk (sclk_global),
13
14
15
16 /*
17 always @(posedge sclk_global)
18
19
        \$fsdbDumpMem (testbench.top.gc.shader\_0.uvector0.umacc\_gpr0.udum\_mem.bram);
20
      end
21 */
22
23 endmodule // sequencer_top
24
25
                                      Page 91 of 92
                                                                                                                                               Page 92 of 92
                                                                     Ex. 2093 - sq.v
                                                                                                                                                                              Ex. 2093 - sq.v
```

```
1 'include "header.v"
                                                                                                           1 // issues:
                                                                                                           2 //-
 4 \hspace{0.5cm} /\hspace{0.1cm}/\hspace{0.1cm} \$ Id: /\hspace{-0.1cm}/\hspace{0.1cm} depot/r 400/devel/parts\_lib/src/gfx/sq/ais/sq\_ais\_output.v \# 16 \,\$
                                                                                                           5 //
                                                                                                           5 'include "sq defs.v"
 6 // $Change: 41217 $
 7 //
                                                                                                           7 module sq_ais_output
 8 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                           8 (
                © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
10 //
                                                                                                          10 aiq0_export_id, //
11 //
               All rights reserved. This notice is intended as a precaution against
                                                                                                          11 aiq0 pc base, // param cache base
                                                                                                          12 aiq0_valid_bits, // valid bits (from ctl packet)
              inadvertent publication and does not imply publication or any waiver
13 //
              of confidentiality. The year included in the foregoing notice is the
                                                                                                          13 aiq0_context_id, // state (context)
14 //
              year of creation of the work.
                                                                                                          14 aiq0_instr, // instruction
15 //
                                                                                                          15 aiq0_gpr_rd_en, //
17 aiq1_export_id, //
18 // sq ais output.v
                                                                                                          18 aig1 pc base, // param cache base
                                                                                                          19 aiq1_valid_bits, // valid bits (from ctl packet)
20 // - takes ALU instruction data from both AIQs
                                                                                                          20 aiq1_context_id,
                                                                                                                                   // state (context)
21 // - selects AIQ output to send to SP based on alu phase
                                                                                                          21 aiq1_instr, // instruction
22 // - selects AIQ constant store read addr to send to ACS based on alu_phase
                                                                                                                aiq1_gpr_rd_en, //
                                                                                                          23
24 // - also takes tex instr's gpr read addr from TIQ
                                                                                                          24 // inputs from the AISs
                                                                                                          25 ais0_acs_rd_rts, // alu const store read addr valid
                                      Page 1 of 34
                                                                                                                                                 Page 2 of 34
                                                             Ex. 2094 - sq ais output.v
                                                                                                                                                                        Ex. 2094 - sq ais output.v
 1 ais0_acs_rd_addr, // alu constant store read address (from instr)
                                                                                                           1 acs_rd_context_id, // alu constant store read context_id
 2 ais0 instr start, // just OR these guys before reg to SP
                                                                                                           2 acs rd data, // alu constant store read data
      ais0_instr_stall,
                                                                                                           4 // outputs to SP
 5 ais1 acs rd rts, // alu const store read addr valid
                                                                                                           5 SQ SP gpr wr addr,
 6 ais1_acs_rd_addr, // alu const store read addr
                                                                                                           6 SQ_SP_gpr_wr_en,
 8 ais1 instr stall,
                                                                                                           8 SQ SP gpr rd en,
                                                                                                                SQ SP gpr phase,
10 // other inputs that get muxed out to the SP or SX
11 tis_gpr_rd_addr, // texture fetch read address
                                                                                                          11 SQ_SP_gpr_channel_mask,
12 tis gpr rd en, // texture fetch read address
                                                                                                          12
13 ia_vertex_sel, // select VISM gpr write address, enable to drive to SP when 1 (select 14 pixel if 0)
                                                                                                          13 SQ_SP_instr_start,
                                                                                                          14 SQ_SP_instr_stall,
15 vi_gpr_wr_addr, // VISM gpr write address
                                                                                                          15 SQ_SP_instr,
16 vi gpr wr en, // VISM gpr write enable
                                                                                                          16
                                                                                                                SQ_SP_const,
      pi_gpr_wr_addr, // PISM gpr write address
                                                                                                          17
18
      pi_gpr_wr_en, // VISM gpr write enable
                                                                                                          18 //
19
                                                                                                          19 SQ_SP_exporting,
                                                                                                          20 SQ_SP_exp_id,
21 gpr_phase,
                    // GPR phase
                                                                                                          21 u0 SO SP write mask.
22 alu_phase,
                     // alu interleaving phase
                                                                                                                u1_SQ_SP_write_mask,
                                                                                                          23 u2_SQ_SP_write_mask,
24 // ALU Constant Store interface
                                                                                                          24 u3 SQ SP write mask,
25 acs rd rts, // alu constant store read addr valid
26 acs_rd_addr, // alu constant store read addr
                                                                                                                                                 Page 4 of 34
                                     Page 3 of 34
                                                            Ex. 2094 - sq ais output.v
                                                                                                                                                                        Ex. 2094 - sq ais output.v
```

ATI 2094 LG v. ATI IPR2015-00326

```
// outputs to SX
                                                                                                                           input [0:0] aiq0_export_id;
2
                                                                                                                     2
      SQ_SX_pc_wr_addr,
       SQ_SX_pc_wr_en,
                                                                                                                           input [6:0] aiq1_pc_base;
       SQ_SX_pc_channel_mask,
                                                                                                                           input [63:0] aiq1_valid_bits;
                                                                                                                           input [2:0] aiq1_context_id;
                                                                                                                           input [101:0] aiq1_instr;
                                                                                                                            input [0:0] aiq1_gpr_rd_en;
                                                                                                                            input [0:0] aiq1_export_id;
10
                                                                                                                           input [0:0] ais0_acs_rd_rts;
       // -- parameters --
11
                                                                                                                           input [8:0] ais0_acs_rd_addr;
      parameter LO = 1'b0;
12
                                                                                                                           input [0:0] ais0_instr_start;
13
      parameter HI = 1'b1;
                                                                                                                    13
                                                                                                                            input [0:0] ais0_instr_stall;
       parameter X = 1'bx;
14
15
                                                                                                                    15
                                                                                                                           input [0:0] ais1 acs rd rts;
16
                                                                                                                    16
                                                                                                                            input [8:0] ais1 acs rd addr;
17
                                                                                                                    17
                                                                                                                            input [0:0] ais1_instr_start;
      // -- ios --
                                                                                                                    18
                                                                                                                            input [0:0] ais1_instr_stall;
18
19
                                                                                                                    19
20
                                                                                                                    20
21
       input [6:0] aiq0_pc_base;
                                                                                                                    21
                                                                                                                           input [6:0] tis_gpr_rd_addr;
22
       input [63:0] aiq0_valid_bits;
                                                                                                                    22
                                                                                                                           input [0:0] tis_gpr_rd_en;
23
       input [2:0] aiq0_context_id;
                                                                                                                    23
                                                                                                                           input [0:0] ia_vertex_sel;
24
       input [101:0] aiq0_instr;
                                                                                                                    24
                                                                                                                           input [6:0] vi_gpr_wr_addr;
       input [0:0] aiq0_gpr_rd_en;
                                                                                                                            input [0:0] vi_gpr_wr_en;
                                           Page 5 of 34
                                                                                                                                                               Page 6 of 34
                                                                   Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                       Ex. 2094 - sq_ais_output.v
       input [6:0] pi_gpr_wr_addr;
                                                                                                                           output [20:0] SQ_SP_instr;
2
       input [3:0] pi_gpr_wr_en;
                                                                                                                           output [127:0] SQ SP const;
                                                                                                                           output [0:0] SQ_SP_exporting;
                                                                                                                           output [0:0] SQ_SP_exp_id;
       input [1:0] gpr_phase;
                                                                                                                           output [3:0] u0_SQ_SP_write_mask;
       input [0:0] alu_phase;
                                                                                                                           output [3:0] u1_SQ_SP_write_mask;
                                                                                                                           output [3:0] u2_SQ_SP_write_mask;
                                                                                                                            output [3:0] u3_SQ_SP_write_mask;
10
       output acs_rd_rts;
                                                                                                                    10
11
       output [8:0] acs_rd_addr;
                                                                                                                    11
                                                                                                                           output [6:0] SQ_SX_pc_wr_addr;
12
       output [2:0] acs_rd_context_id;
                                                                                                                    12
                                                                                                                           output [0:0] SQ_SX_pc_wr_en;
13
       input [127:0] acs_rd_data;
                                                                                                                           output [3:0] SQ_SX_pc_channel_mask;
14
15
                                                                                                                    15
16
       output [6:0] SQ_SP_gpr_wr_addr;
                                                                                                                          reg [6:0] SQ_SP_gpr_wr_addr;
17
      output [3:0] SQ_SP_gpr_wr_en;
                                                                                                                          reg [3:0] SQ_SP_gpr_wr_en;
18
      output [6:0] SQ_SP_gpr_rd_addr;
                                                                                                                    18
                                                                                                                          reg [6:0] SQ_SP_gpr_rd_addr;
19
      output [0:0] SQ_SP_gpr_rd_en;
                                                                                                                           reg [0:0] SQ_SP_gpr_rd_en;
20
      output [1:0] SQ SP gpr phase;
                                                                                                                    20
                                                                                                                           reg [1:0] SQ SP gpr phase;
21
      output [1:0] SQ SP gpr input sel;
                                                                                                                    21
                                                                                                                           reg [1:0] SQ SP gpr input sel;
22
                                                                                                                    22
      output [3:0] SQ_SP_gpr_channel_mask;
                                                                                                                           reg [3:0] SQ_SP_gpr_channel_mask;
23
                                                                                                                    23
24
      output [0:0] SQ_SP_instr_start;
                                                                                                                    24
                                                                                                                          reg [0:0] SQ_SP_instr_start;
25
       output [0:0] SQ_SP_instr_stall;
                                                                                                                           reg [0:0] SQ_SP_instr_stall;
                                          Page 7 of 34
                                                                                                                                                               Page 8 of 34
                                                                  Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                       Ex. 2094 - sq_ais_output.v
```

```
reg [20:0] SQ SP instr;
                                                                                                                                        reg [7:0] isr_scalar_dest_q;
 2
        reg [127:0] SQ_SP_const;
                                                                                                                                        reg [3:0] isr_vector_mask_q;
                                                                                                                                        reg [3:0] isr_scalar_mask_q;
        reg [0:0] SQ_SP_exporting;
                                                                                                                                        reg [1:0] isr_pred_sel_q;
        reg [0:0] SQ_SP_exp_id;
                                                                                                                                         reg [6:0] isr_pc_base_q;
        reg [3:0] u0_SQ_SP_write_mask;
                                                                                                                                         reg [0:0] isr_instr_stall_q;
        reg [3:0] u1_SQ_SP_write_mask;
        reg [3:0] u2_SQ_SP_write_mask;
                                                                                                                                        reg [7:0] isr_vector_dest_q1;
        reg [3:0] u3_SQ_SP_write_mask;
                                                                                                                                        reg [7:0] isr_scalar_dest_q1;
                                                                                                                                        reg [3:0] isr_vector_mask_q1;
11
        reg [6:0] SQ_SX_pc_wr_addr;
                                                                                                                                       reg [3:0] isr_scalar_mask_q1;
12
        reg [0:0] SQ_SX_pc_wr_en;
                                                                                                                                       reg [1:0] isr_pred_sel_q1;
13
        reg [3:0] SQ_SX_pc_channel_mask;
                                                                                                                                        reg [6:0] isr_pc_base_q1;
14
                                                                                                                                        reg [0:0] isr_instr_stall_q1;
15
                                                                                                                                15
        input clk;
16
                                                                                                                                16
        input reset:
                                                                                                                                        //wire scalar export;
17
                                                                                                                                17
                                                                                                                                        wire scalar_export_pc;
18
                                                                                                                                18
                                                                                                                                        //wire vector export;
19
                                                                                                                                19
                                                                                                                                        wire vector export pc;
20
        // -- internal signals --
                                                                                                                                20
                                                                                                                                         wire export;
21
                                                                                                                                21
                                                                                                                                         wire export_pc;
22
                                                                                                                                22
23
        // ISR - save this part of instruction when IQ is popped
                                                                                                                                23
24
        // - needed for GPR result write and exports to PC
                                                                                                                                24
        reg [7:0] isr_vector_dest_q;
                                                                                                                                        // -- module instatiations --
                                               Page 9 of 34
                                                                                                                                                                               Page 10 of 34
                                                                          Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                                          Ex. 2094 - sq_ais_output.v
                                                                                                                                        assign vector export pc = export & ~[(isr vector dest q1[5:4]);
                                                                                                                                        assign export_pc = scalar_export_pc | vector_export_pc;
       // -- combinational logic --
                                                                                                                                         // -- registers --
        // - mux ACS read addr based on alu phase
        \ensuremath{//} - NOTE the addr is the opposite phase since const store read starts 4 cycles before the instr
10
                                                                                                                                10
11
                                                                                                                                11
                                                                                                                                        // -- Instruction Input Staging Register --
12
        assign acs_rd_addr = alu_phase ? ais0_acs_rd_addr : ais1_acs_rd_addr;
                                                                                                                                12
        assign acs_rd_context_id = alu_phase ? aiq0_context_id : aiq1_context_id;
                                                                                                                                       /\!/ - holds the instruction data from the AIQ for use by GPR and PC writes (is reloaded by other thread
        //assign acs_rd_rts = alu_phase ? aisl_acs_rd_rts : ais0_acs_rd_rts;
15
        assign acs_rd_rts = ais1_acs_rd_rts | ais0_acs_rd_rts;
                                                                                                                                        // before GPR and PC writes occur, so relavent info must be kept here)
16
                                                                                                                                        // - need to save stall to know whether to assert WE to gprs or PC also
17
                                                                                                                                        // - must reload after every instruction even if AIS is idle to get the stall info saved
18
        // - decode ISR instruction info for GPR writeback and Param Cache Writes
                                                                                                                                        // - actually need two stages here since the AIQ must be popped for the next constant access
19
                                                                                                                                19
        /\!/ assign\ scalar\_export = \sim \! isr\_pred\_sel\_q1[1]\ \&\ isr\_scalar\_dest\_q1[7];
20
                                                                                                                                20
                                                                                                                                         always @(posedge clk)
21
        /\!/ assign\ vector\_export = \sim \!\! isr\_pred\_sel\_ql[1]\ \&\ isr\_vector\_dest\_ql[7];
                                                                                                                                21
                                                                                                                                         begin
22
        /\!/ assign\ scalar\_export = isr\_scalar\_dest\_q1[7];
                                                                                                                                22
                                                                                                                                              if (reset)
23
        //assign vector_export = isr_scalar_dest_q1[7];
                                                                                                                                23
24
        assign export = isr_scalar_dest_q1[7];
                                                                                                                                24
                                                                                                                                              // stall forces a NOP to the shader pipe
25
                                                                                                                                25
                                                                                                                                                 // - all instruction bits are don't care when stall == 1, so they don't need to be reset
        assign\ scalar\_export\_pc = export\ \&\ \sim |(isr\_scalar\_dest\_q1[5:4]);
                                                                                                                                                 // - stall forces WE to GPR and PC to be deasserted
                                              Page 11 of 34
                                                                                                                                                                              Page 12 of 34
                                                                         Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                                          Ex. 2094 - sq_ais_output.v
```

```
//isr scalar dest q <= 0;
                                                                                                                                     isr_vector_mask_q <= aiq1_instr[19:16];
 2
             //isr_scalar_mask_q <= 0;
                                                                                                                                     isr\_pred\_sel\_q \quad \mathop{<=} aiq1\_instr[60:59];
             /\!/isr\_vector\_dest\_q <= 0;
                                                                                                                                     isr\_pc\_base\_q \quad <= aiql\_pc\_base;
             //isr_vector_mask_q <= 0;
                                                                                                                                     isr\_instr\_stall\_q \mathrel{<=} ais1\_instr\_stall;
             /\!/isr\_pred\_sel\_q \quad <= 0;
             //isr\_pc\_base\_q <= 0;
              isr_instr_stall_q \le HI;
                                                                                                                                     isr_scalar_dest_q <= isr_scalar_dest_q;
10
             else if ( (gpr_phase == 2'b11) & (alu_phase == LO) ) //(ais0_ld_isr)
                                                                                                                                       isr\_vector\_dest\_q \mathrel{<=} isr\_vector\_dest\_q;
11
                                                                                                                                    isr_scalar_mask_q <= isr_scalar_mask_q;
12
             isr\_scalar\_dest\_q \mathrel{<=} aiq0\_instr[15:8];
                                                                                                                                      isr_vector_mask_q <= isr_vector_mask_q;
13
             isr_vector_dest_q <= aiq0_instr[ 7:0];
                                                                                                                                     isr_pred_sel_q <= isr_pred_sel_q;
14
             isr_scalar_mask_q <= aiq0_instr[23:20];
                                                                                                                                     isr_pc_base_q <= isr_pc_base_q;
15
                                                                                                                        15
             isr vector mask q <= aiq0 instr[19:16];
                                                                                                                                     isr instr stall q <= isr instr stall q;
16
             isr pred sel q <= aiq0 instr[60:59];
                                                                                                                        16
             isr_pc_base_q <= aiq0_pc_base;
17
                                                                                                                        17
                                                                                                                                end
18
             isr_instr_stall_q <= ais0_instr_stall;
                                                                                                                        18
19
                                                                                                                        19
20
                                                                                                                        20
                                                                                                                               // ISR1 - need to pipe ISR0 to keep it around for the GPR/PC write
21
             else if ( (gpr_phase == 2'b11) & (alu_phase == HI) ) //(ais1_ld_isr)
                                                                                                                        21
22
                                                                                                                        22
                                                                                                                               always @(posedge clk)
23
             isr\_scalar\_dest\_q \mathrel{<=} aiq1\_instr[15:8];
                                                                                                                        23
24
             isr_vector_dest_q <= aiq1_instr[ 7:0];
                                                                                                                        24
                                                                                                                                    //if (reset)
25
             isr_scalar_mask_q <= aiq1_instr[23:20];
                                                                                                                        25
                                                                                                                                     //begin
                                           Page 13 of 34
                                                                                                                                                                   Page 14 of 34
                                                                     Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                             Ex. 2094 - sq_ais_output.v
             //isr_scalar_dest_q <= 0;
                                                                                                                                       isr_vector_mask_q1 <= isr_vector_mask_q1;
 2
                                                                                                                         2
             //isr scalar mask q <= 0;
                                                                                                                                     isr_pred_sel_ql <= isr_pred_sel_ql;
             //isr vector dest q <= 0;
                                                                                                                                     isr_pc_base_ql <= isr_pc_base_ql;
             //isr vector mask q <= 0;
                                                                                                                                     isr_instr_stall_q1 <= isr_instr_stall_q1;
             //isr_pred_sel_q <= 0;
                                                                                                                                      end
             //isr_pc_base_q <= 0;
                                                                                                                         6
                                                                                                                                end
              /\!/isr\_instr\_stall\_q \mathrel{<=} HI;
                                                                                                                         7
10
            if \, (\, (gpr\_phase == 2'b11) \, )
                                                                                                                        10
                                                                                                                               // -- SP instruction, write_mask --
11
                                                                                                                        11
12
             isr_scalar_dest_q1 <= isr_scalar_dest_q;
                                                                                                                        12
                                                                                                                               // - valid with instruction start
13
               isr_vector_dest_q1 <= isr_vector_dest_q;
                                                                                                                        13
14
             isr_scalar_mask_q1 <= isr_scalar_mask_q;
                                                                                                                        14
                                                                                                                               always @(posedge clk)
15
              isr_vector_mask_q1 <= isr_vector_mask_q;
                                                                                                                        15
             isr_pred_sel_ql <= isr_pred_sel_q;
                                                                                                                               case (gpr_phase)
17
             isr_pc_base_q1 <= isr_pc_base_q;
                                                                                                                                `SQ_SRCB_PHASE: begin
18
             isr_instr_stall_ql \le isr_instr_stall_q;
                                                                                                                                 case (alu_phase)
19
                                                                                                                                    LO: begin
                                                                                                                        20
21
                                                                                                                            22
23
22
23
             isr scalar dest q1 <= isr scalar dest q1;
                                                                                                                             u2\_SQ\_SP\_write\_mask <= aiq0\_valid\_bits ~ [11:8]; ~ u3\_SQ\_SP\_write\_mask <= aiq0\_valid\_bits ~ [15:12];
24
              isr vector dest q1 <= isr vector dest q1;
25
             isr\_scalar\_mask\_q1 \mathrel{<=} isr\_scalar\_mask\_q1;
                                           Page 15 of 34
                                                                                                                                                                   Page 16 of 34
                                                                     Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                             Ex. 2094 - sq_ais_output.v
```

```
HI: begin
 1
                                                                                                                                    end
     SQ_SP_instr aiq1_instr[101:99]};
                                                                                                                           2
                           <= {aiq1_instr[07:00], aiq1_instr[55:48], aiq1_instr[58],
                                                                                                                                    'SQ FA PHASE: begin
                                                                                                                                     case (alu_phase)
     LO: begin
      u2\_SQ\_SP\_write\_mask <= aiq1\_valid\_bits \ [11:8]; \ u3\_SQ\_SP\_write\_mask <= aiq1\_valid\_bits \ [15:12]; 
                                                                                                                                                     <= {aiq0_instr[23:16], aiq0_instr[39:32], aiq0_instr[56],
                                                                                                                                 u0\_SQ\_SP\_write\_mask <= aiq0\_valid\_bits ~ [35:32]; ~ u1\_SQ\_SP\_write\_mask <= aiq0\_valid\_bits ~ [39:36]; 
           end
          endcase
                                                                                                                               u2_SQ_SP_write_mask <= aiq0_valid_bits [43:40]; u3_SQ_SP_write_mask <= aiq0_valid_bits [47:44];
10
                                                                                                                          11
                                                                                                                                      end
11
          'SQ SRCC PHASE: begin
12
                                                                                                                          12
                                                                                                                                       HI: begin
          case (alu phase)
                                                                                                                               \label{eq:sq_sp_instr} SQ\_SP\_instr <= \{aiq1\_instr[23:16], \quad aiq1\_instr[39:32], \quad aiq1\_instr[56], \\ aiq1\_instr[95:93]\};
13
            LO: begin
     SQ_SP_instr
aiq0_instr[98:96]};
                            <= {aiq0_instr[15:08], aiq0_instr[47:40], aiq0_instr[57],
                                                                                                                                \label{eq:condition} $$u0_SQ_SP\_write\_mask <= aiq0\_valid\_bits $$[19:16]; $$u1_SQ_SP\_write\_mask <= aiq0\_valid\_bits $$[23:20];
                                                                                                                                u2\_SQ\_SP\_write\_mask \le aiq1\_valid\_bits [43:40]; u3\_SQ\_SP\_write\_mask \le aiq1\_valid\_bits [47:44];
     19
                                                                                                                                      end
20
           end
                                                                                                                          20
                                                                                                                                      endcase
21
             HI: begin
                                                                                                                          21
                                                                                                                                    end
     SQ\_SP\_instr <= \{aiq1\_instr[15:08], \quad aiq1\_instr[47:40], \quad aiq1\_instr[57], \\ aiq1\_instr[98:96]\};
                                                                                                                          22
22
23
                                                                                                                                    'SQ SRCA PHASE: begin
                                                                                                                          23
                                                                                                                                    case (alu_phase)
     \label{eq:condition} $$u0_SQ_SP\_write\_mask <= aiq1\_valid\_bits \ [19:16]; \ u1\_SQ\_SP\_write\_mask <= aiq1\_valid\_bits \ [23:20];
24
25
                                                                                                                          24
                                                                                                                                       LO: begin
     \begin{array}{lll} SQ\_SP\_instr & <= & \{aiq0\_instr[23:16], & aiq0\_instr[25:24], & aiq0\_instr[31:26], \\ aiq0\_instr[92:88]\}; \end{array}
                                                                                                                               28
           end
           endcase
                                            Page 17 of 34
                                                                                                                                                                       Page 18 of 34
                                                                      Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                                 Ex. 2094 - sq_ais_output.v
     \label{eq:condition} \begin{array}{lll} u2\_SQ\_SP\_write\_mask & <= & aiq0\_valid\_bits & [59:56]; & u3\_SQ\_SP\_write\_mask & <= & aiq0\_valid\_bits & [63:60]; \\ \end{array}
                                                                                                                                      SQ SP instr start <= ais0 instr start;
                                                                                                                           2
                                                                                                                                      SQ SP instr stall <= ais0 instr stall;
           end
                                                                                                                                      /\!/ \ logic \ for \ exporting = \sim pred\_sel[1] \ \& \ (scalar\_dest[7] \ | \ vector\_dest[7])
             HI: begin
                                                                                                                                      // new logic for exporting = scalar_dest[7]
            SQ\_SP\_instr \leftarrow \{aiq1\_instr[23:16], aiq1\_instr[25:24], aiq1\_instr[31:26],
                                                                                                                                      // remove this stall mux
     \label{eq:u0_SQ_SP_write_mask} <= aiq1\_valid\_bits ~ [51:48]; ~ u1\_SQ\_SP\_write\_mask ~ = aiq1\_valid\_bits [55:52];
                                                                                                                                        if (ais0 instr stall)
                                                                                                                           7
                                                                                                                                       SQ\_SP\_exporting \le LO;
     \label{eq:condition} \begin{array}{lll} u2\_SQ\_SP\_write\_mask & <= & aiq1\_valid\_bits & [59:56]; & u3\_SQ\_SP\_write\_mask & <= & aiq1\_valid\_bits & [63:60]; \\ \end{array}
                                                                                                                                       /\!/SQ\_SP\_exporting <= \sim\! aiq0\_instr[60] \ \& \ (aiq0\_instr[15] \ | \ aiq0\_instr[7]);
11
           end
12
                                                                                                                          10
                                                                                                                                       SQ\_SP\_exporting \  \, <= aiq0\_instr[15];
           endcase
                                                                                                                          11
                                                                                                                                      SQ_SP_exp_id <= aiq0_export_id; //
13
          end
14
                                                                                                                          12
         endcase
                                                                                                                                    HI: // if interleaving is disabled, need to make sure controls are not driven
                                                                                                                          13
15
        end
                                                                                                                          14
16
                                                                                                                          15
                                                                                                                                      SQ_SP_instr_start <= ais1_instr_start;
17
                                                                                                                                      SQ\_SP\_instr\_stall <= ais1\_instr\_stall;
18
       // SQ SP instr start
                                                                                                                                      // remove this stall mux
19
       // SQ_SP_instr_stall
                                                                                                                                       if ( ais l_instr_stall ) SQ_SP_exporting <= LO;
20
       // SQ_SP_exporting
                                                                                                                                                    SQ_SP_exporting <= aiq1_instr[15];
21
       // SQ_SP_exp_id
                                                                                                                          20
                                                                                                                                      SQ SP exp id <= aiq1 export id; //
22
       //
                                                                                                                          21
                                                                                                                                      end
23
        always @(posedge clk)
                                                                                                                          22
24
                                                                                                                                    endcase
        begin
                                                                                                                          23
25
                                                                                                                                   end
         case (alu_phase)
                                                                                                                          24
26
         LO:
           begin
                                                                                                                          25
                                            Page 19 of 34
                                                                                                                                                                       Page 20 of 34
                                                                      Ex. 2094 - sq ais output.v
                                                                                                                                                                                                 Ex. 2094 - sq_ais_output.v
```

```
// SQ SP gpr phase
                                                                                                                                 end
                                                                                                                          2 */
 2
      // SQ_SP_gpr_input_sel
 3
                                                                                                                          3
 4
      always @(posedge clk)
                                                                                                                          4 always @(posedge clk)
                                                                                                                                 case (gpr_phase)
        SQ_SP_gpr_phase <= gpr_phase;
      SQ\_SP\_gpr\_input\_sel <= \{ia\_vertex\_sel, ~ia\_vertex\_sel\}; \# 00: cnt, \ 01: pix, \ 10: vtx \ (fix needed for count)
                                                                                                                                  `SQ_SRCA_PHASE: begin
                                                                                                                                                                              // have to invert this to get the srcA addr in
                                                                                                                              case (~alu_phase)
a cycle early
10
                                                                                                                                     LO: SQ_SP_gpr_rd_addr <= aiq0_instr[86:80];
11
                                                                                                                         11
                                                                                                                                      HI: SQ_SP_gpr_rd_addr <= aiq1_instr[86:80];
12
13
      // -- SP gpr read address, read enable --
                                                                                                                                                                              // have to invert this to get the srcA addr in
                                                                                                                             case (~alu_phase)
a cycle early
14
                                                                                                                         15
                                                                                                                                     LO: SQ_SP_gpr_rd_en <= aiq0_gpr_rd_en;
15
      // - the read address comes directly from the ALU or Texture Instruction Queue (IQ)
                                                                                                                         16
                                                                                                                                     HI: SQ\_SP\_gpr\_rd\_en \mathrel{<=} aiql\_gpr\_rd\_en;
       // - the read address was calculated prior to being loaded into the IQ
                                                                                                                         17
                                                                                                                                   endcase
17
       // - the read enable is just the RTS out of the IQ
                                                                                                                         18
18 /*
                                                                                                                         19
                                                                                                                                   `SQ_SRCB_PHASE: begin
19
       reg [0:0] aiq_gpr_rd_en;
                                                                                                                         20
20
                                                                                                                         21
                                                                                                                                     LO: SQ_SP_gpr_rd_addr <= aiq0_instr[78:72];
21
       always @(alu_phase or aiq0_gpr_rd_en or aiq1_gpr_rd_en)
                                                                                                                         22
                                                                                                                                     HI: SQ_SP_gpr_rd_addr <= aiql_instr[78:72];
22
23
          case (alu_phase)
                                                                                                                                   case (alu_phase)
24
            LO: aiq_gpr_rd_en = aiq0_gpr_rd_en;
                                                                                                                                     LO: SQ_SP_gpr_rd_en <= aiq0_gpr_rd_en;
25
             HI: aiq_gpr_rd_en = aiql_gpr_rd_en;
                                                                                                                                      HI: SQ_SP_gpr_rd_en <= aiq1_gpr_rd_en;
                                           Page 21 of 34
                                                                                                                                                                    Page 22 of 34
                                                                     Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                              Ex. 2094 - sq_ais_output.v
           endcase
 2
                                                                                                                          2
          end
                                                                                                                                always @(posedge clk)
          `SQ_SRCC_PHASE: begin
           case (alu_phase)
                                                                                                                                  SQ SP gpr wr en <= 0; // clear WE by default
             LO: SQ\_SP\_gpr\_rd\_addr <= aiq0\_instr[70:64];
                                                                                                                                  case (gpr_phase)
                                                                                                                                  'SO PS PHASE:
             HI: SQ\_SP\_gpr\_rd\_addr \mathrel{<=} aiql\_instr[70:64];
           endcase
           case (alu_phase)
                                                                                                                                     SQ\_SP\_gpr\_wr\_addr <= isr\_scalar\_dest\_q1[6:0];
             LO: SQ_SP_gpr_rd_en <= aiq0_gpr_rd_en;
                                                                                                                                     SQ_SP_gpr_channel_mask <= isr_scalar_mask_q1;
10
             HI: SQ_SP_gpr_rd_en <= aiq1_gpr_rd_en;
                                                                                                                         10
                                                                                                                                     //if ( ~scalar_export & ~isr_instr_stall_q1)
11
           endcase
                                                                                                                         11
                                                                                                                                     if ( ~export & ~isr_instr_stall_q1)
                                                                                                                             SQ_SP_gpr_wr_en <= 4'b1111;
exporting and not stalling
12
                                                                                                                                                                              // assert gpr write enable when not
13
          `SQ_FA_PHASE: begin
                                                                                                                         14
14
            SQ_SP_gpr_rd_addr <= tis_gpr_rd_addr;
                                                                                                                         15
                                                                                                                                   'SQ_PV_PHASE:
15
           SQ_SP_gpr_rd_en <= tis_gpr_rd_en;
16
                                                                                                                         17
                                                                                                                                       SQ SP gpr wr addr <= isr vector dest q1[6:0]:
17
         endcase
                                                                                                                                     SQ SP gpr channel mask <= isr vector mask q1;
18
                                                                                                                                     if (~export & ~isr instr stall q1)
                                                                                                                         19
19
                                                                                                                                      SQ_SP_gpr_wr_en <= 4'b1111;
                                                                                                                         20
20
                                                                                                                         21
21
                                                                                                                         22
                                                                                                                                   'SQ ID PHASE:
22
       // -- SP gpr write address, write enable, channel mask --
                                                                                                                         23
23
                                                                                                                         24
                                                                                                                                     case (ia_vertex_sel)
24
       // - don't have to use alu phase as a select here since it was used to load ISR
                                                                                                                         25
                                                                                                                                       LO: begin
       // - the write address was calcualted prior to being loaded into the Instruction Queue
                                                                                                                                       SQ\_SP\_gpr\_wr\_addr <= pi\_gpr\_wr\_addr;
                                           Page 23 of 34
                                                                                                                                                                   Page 24 of 34
                                                                     Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                              Ex. 2094 - sq_ais_output.v
```

```
1
              SQ_SP_gpr_wr_en <= pi_gpr_wr_en;
                                                                                                                                   reg [6:0] pc_wr_addr_q1;
 2
                end
                                                                                                                                   reg [6:0] pc_wr_addr_q2;
              HI: begin
                                                                                                                                   reg pc_wr_en_q;
              SQ\_SP\_gpr\_wr\_addr <= vi\_gpr\_wr\_addr;
                                                                                                                                   reg pc_wr_en_ql;
             SQ\_SP\_gpr\_wr\_en \ <= \{4\{vi\_gpr\_wr\_en\}\};
                                                                                                                                   reg pc_wr_en_q2;
                                                                                                                                   reg [3:0] pc_channel_mask_q;
                                                                                                                                   reg [3:0] pc_channel_mask_q1;
             SQ_SP_gpr_channel_mask <= 4'b1111;
                                                                                                                                   reg [3:0] pc_channel_mask_q2;
10
          SQ_FD_PHASE: begin
                                                                                                                                   wire ld_pc_wr_addr = (gpr_phase == `SQ_PS_PHASE);
11
             SQ\_SP\_gpr\_wr\_addr <= 7"b0;
                                                                                                                           11
12
             SQ_SP_gpr_wr_en <= 4'b0;
                                                                                                                           12
                                                                                                                                   always @(posedge clk)
13
           SQ_SP_gpr_channel_mask <= 4'b0;
                                                                                                                           13
14
                                                                                                                            14
                                                                                                                                         case ( ld_pc_wr_addr )
15
                                                                                                                           15
         endcase
16
                                                                                                                            16
        end
                                                                                                                                       begin
17
                                                                                                                            17
                                                                                                                                           case ( scalar export pc )
18
                                                                                                                            18
19
                                                                                                                            19
20
       // -- PC Write Address Register --
                                                                                                                            20
                                                                                                                                          pc\_wr\_addr\_q \mathrel{<=} isr\_pc\_base\_q1 + isr\_scalar\_dest\_q1[5:0];
21
                                                                                                                           21
                                                                                                                                          pc\_channel\_mask\_q \mathrel{<=} isr\_scalar\_mask\_q1;
    // - load when phase is PS (this lines up with the ISR load and IQ pop, i.e. the last cycle the data
                                                                                                                           22
22
23
                                                                                                                           23
                                                                                                                                          LO:
24
       // from the IQ is valid for this instruction)
                                                                                                                           24
25
                                                                                                                           25
                                                                                                                                          pc\_wr\_addr\_q \mathrel{<=} isr\_pc\_base\_q1 + isr\_vector\_dest\_q1[5:0];
       reg [6:0] pc_wr_addr_q;
                                            Page 25 of 34
                                                                                                                                                                        Page 26 of 34
                                                                       Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                                   Ex. 2094 - sq_ais_output.v
               pc channel mask q <= isr vector mask q1;
 2
                                                                                                                             2
             end
                                                                                                                                        pc_wr_en_q1 <= pc_wr_en_q;
                endcase
                                                                                                                                        pc\_wr\_en\_q2 \mathrel{<=} pc\_wr\_en\_q1;
 4
            end
                                                                                                                                        SQ_SX_pc_wr_en <= pc_wr_en_q2;
 5
          LO:
                                                                                                                                      pc\_channel\_mask\_q1 \mathrel{<=} pc\_channel\_mask\_q;
               case ( pc_wr_en_q )
                                                                                                                                      pc\_channel\_mask\_q2 <= pc\_channel\_mask\_q1;
               HI \colon pc\_wr\_addr\_q \mathrel{<=} pc\_wr\_addr\_q + 1;
                                                                                                                                      SQ\_SX\_pc\_channel\_mask <= pc\_channel\_mask\_q2;
                LO: pc\_wr\_addr\_q <= pc\_wr\_addr\_q;
                                                                                                                            9
10
                                                                                                                           10
11
             pc\_channel\_mask\_q \mathrel{<=} pc\_channel\_mask\_q;
                                                                                                                           11
12
                                                                                                                           12
                                                                                                                                   // ALU Constant Store read data
13
             endcase // case( ld_pc_wr_addr )
                                                                                                                           13
14
                                                                                                                                    always @(posedge clk)
15
16
                                                                                                                                       SQ_SP_const = acs_rd_data;
17
                                                                                                                           17
       // SX param cache write address
18
       // SX param cache write enable
                                                                                                                           18
19
                                                                                                                            19
       // SX param cache channel mask
20
       // - these guys need to be delayed 2 cycles due to the block interface delay btwn SP and SX
                                                                                                                           20
                                                                                                                                  // misc interface registers
21
                                                                                                                           21 /*
       always @(posedge clk)
22
                                                                                                                           22
                                                                                                                                  wire ais fetch stall = tp fetch stall; // need to add the correct timing for ais fetch stall
23
             pc\_wr\_addr\_q1 \mathrel{<=} pc\_wr\_addr\_q;
                                                                                                                           23
                                                                                                                                                           // why does it not go directly from TPC to SPs?
24
             pc\_wr\_addr\_q2 \mathrel{<=} pc\_wr\_addr\_q1;
                                                                                                                           24
                                                                                                                                   always @(posedge clk)
25
             SQ\_SX\_pc\_wr\_addr \mathrel{<=} pc\_wr\_addr\_q2; \qquad \textit{// these all need to go back to } q1 \ldots
                                                                                                                            25
                                                                                                                                    begin
                                            Page 27 of 34
                                                                                                                                                                        Page 28 of 34
                                                                       Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                                   Ex. 2094 - sq_ais_output.v
```

```
1
            tp_fetch_stall <= TP_SQ_fetch_stall;
                                                                                                                                      pc_wr_en_q <= LO;
 2
                                                                                                                      2
        end
                                                                                                                                  endcase
 3
                                                                                                                      3
                                                                                                                             end
                                                                                                                           // Parameter Cache Write State Machine:
11
      always @(posedge clk)
                                                                                                                          // - load either scalar or vector param cache address
12
                                                                                                                            // - increment param cache address based on gpr phase and whether scalar or vector is
13
            if (reset)
                                                                                                                     14
                                                                                                                           // to the param cache (can't both export to param cache in same instruction)
14
             pc\_wr\_en\_q \mathrel{<=} LO;
                                                                                                                     15 /*
15
                                                                                                                     16
                                                                                                                            parameter PCW0 = 2'b00;
16
             case ( pc_wr_en_q )
                                                                                                                     17
                                                                                                                           parameter PCW1 = 2'b01:
17
                                                                                                                           parameter PCW2 = 2'b10;
18
19
               // - set when loading pc_wr_addr and we're exporting to the PC and we're not
      stalling
                                                                                                                    19
                                                                                                                            parameter PCW3 = 2'b11;
20
                // - ld_pc_wr_addr is asserted based on phase above for PC write addr register
                                                                                                                    20
21
               if ( ld_pc_wr_addr & export_pc & \sim\!\! isr_instr_stall_q1 )
                                                                                                                           reg [1:0] pcw_current_state;
22
                pc_wr_en_q <= HI;
                                                                                                                    22
                                                                                                                           reg [1:0] pcw_next_state;
23
                                                                                                                    23
               // - clear when loading pc_wr_addr and we're not exporting to the PC or we're
                                                                                                                    24
                                                                                                                           // reg'd outputs
                                                                                                                    25
26
                if ( ld_pc_wr_addr & (~export_pc | isr_instr_stall_q1) )
                                                                                                                           // un-reg'd (combinatorial) outputs
                                          Page 29 of 34
                                                                                                                                                               Page 30 of 34
                                                                   Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                        Ex. 2094 - sq_ais_output.v
                                                                                                                                pcw_next_state = PCW0;
      //reg inc pc wr addr;
 2
                                                                                                                      2
                                                                                                                              next_ais_start = LO;
       //reg ld isr;
 4
                                                                                                                               ais rtr = LO:
                                                                                                                                                   // this guy pops the AIQ
      // state and output registers
 5
      always @(posedge clk)
                                                                                                                                ld isr = LO;
                                                                                                                                                 // this guy loads the ISR
        begin
 7
                                                                                                                      7
            if (reset)
                                                                                                                                case (pcw_current_state)
                                                                                                                      8
                                                                                                                                  PCW0:
              pcw_current_state <= PCW0;
                                                                                                                      9
10
            ais_start <= LO;
                                                                                                                                    // - wait until the AIQ output is valid and the GPR read phase is at the Fetch Address
11
                                                                                                                     12
                                                                                                                                    // - the GPR read addr goes from the AIQ to the output phase mux, and is selected
12
                                                                                                                     13
                                                                                                                                    // when phase == `SQ_FA
13
                                                                                                                                    // - the aluconst read address also goes directly from the AIQ
14
             pcw_current_state <= pcw_next_state;
                                                                                                                     15
15
            ais_start <= next_ais_start;
                                                                                                                                   if (aiq_rts & (gpr_phase == `SQ_FA_PHASE))
16
           end
                                                                                                                     17
                                                                                                                                    begin
17
         end
                                                                                                                                    pcw next state = PCW1;
18
                                                                                                                     19
                                                                                                                                  end
19
      // next state logic
                                                                                                                     20
                                                                                                                                  end
20
      always @(
                                                                                                                     21
21
           aiq_rts or gpr_phase or tp_fetch_stall or
                                                                                                                    22
                                                                                                                                  PCW1:
22
              pcw_current_state
                                                                                                                    23
                                                                                                                                begin
23
                                                                                                                    24
24
       begin
                                                                                                                    25
                                                                                                                                   pcw_next_state = PCW2;
25
           // default assignments
                                                                                                                     26
                                          Page 31 of 34
                                                                                                                                                               Page 32 of 34
                                                                   Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                       Ex. 2094 - sq_ais_output.v
```

```
1
2
             PCW2:
                                                                                                                        2
           begin
              // -
                                                                                                                        4 endmodule
              pcw_next_state = PCW3;
             end
                                                                                                                         6
           begin
              // - kick off the interface state machine
10
                                                                                                                        10
11
              // - load the TP interface data staging register
                                                                                                                        11
              \ensuremath{/\!/} - when the TP stalls, keep sending the same instruction
12
                                                                                                                        12
13
              // - i.e. don't pop the AIQ (via ais_rtr)
                                                                                                                        13
14
                                                                                                                        14
15
              next_ais_start = HI;
                                                                                                                        15
16
              ld_isr = HI;
                                                                                                                        16
17
                                                                                                                        17
18
              if (~tp_fetch_stall) ais_rtr = HI;
                                                                                                                        18
19
                                                                                                                        19
              pcw_next_state = PCW0;
                                                                                                                       20
20
21
22
23
            endcase // case(pcw_current_state)
24
        end // always @ (*)
25
        // - end aiq read state machine
                                           Page 33 of 34
                                                                                                                                                                   Page 34 of 34
                                                                    Ex. 2094 - sq_ais_output.v
                                                                                                                                                                                            Ex. 2094 - sq_ais_output.v
```

```
1 'include "header.v"
                                                                                                                   1 //
                                                                                                                   2 // issues:
 2 //-----
                                                                                                                   3 // - relative mode not implemented yet
 4 \hspace{0.5cm} // \hspace{0.1cm} \$ Id: // depot/r 400/ devel/parts\_lib/src/gfx/sq/ais/sq\_alu\_instr\_queue.v \# 14 \hspace{0.1cm} \$
                                                                                                                   4 // - could this just be a ping-pong buffer?
 5 //
                                                                                                                   5 // - does context id id need to go into the IQ? (currently it does, but can think of no use)
 6 // $Change: 41796 $
 7 //
                                                                                                                   8 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                                   8 'include "sq defs.v"
                  © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
10 //
                                                                                                                   10 module sq_alu_instr_queue
11 //
                All rights reserved. This notice is intended as a precaution against
                                                                                                                  11 (
                inadvertent publication and does not imply publication or any waiver
                                                                                                                  12 write_rts,
13 //
                of confidentiality. The year included in the foregoing notice is the
                                                                                                                  13 write_rtr,
14 //
               year of creation of the work.
                                                                                                                  14
15 //
                                                                                                                  15 // inputs from AIF (ALU Instruction Fetch)
                                                                                                                         aif_export_info, // {export_id, pulse_sx}
17 aif_pc_base_q, // param cache base addr
18 // sq alu instr queue.v
                                                                                                                  18 aif last in group q, // last instruction in series of instructions (done sent back fron ais)
                                                                                                                   19 aif last in thread_q,// last instruction in the whole shader program (no done sent back fron as)
20 ais)
20 // Alu Instruction Queue
                                                                                                                  21 aif_thread_type_q, // vector type (0: pixel, 1: vertex)
21 //
                                                                                                                  22 aif_thread_id_q, // thread id
22 // - instantiates fifo_regs and a simple fifo controller for the queue
                                                                                                                  23
                                                                                                                         aif_ctl_pkt_q, // control packet (valid_bits, gpr_base, context_id)
23 // - calculates source and destination GPR addresses from gpr_base, which is part
                                                                                                                  24
                                                                                                                        aif_instr_q,
                                                                                                                                           // instruction register (registered read from IS - 96 bits)
24 // of the input ctl packet, and from offset in the instruction (absoulte mode)
                                                                                                                  25
25 // or from offset and loop index (relative mode)
                                          Page 1 of 12
                                                                                                                                                             Page 2 of 12
                                                            Ex. 2095 - sq alu instr queue.v
                                                                                                                                                                               Ex. 2095 - sq alu instr queue.v
 1 read_rtr,
                                                                                                                   2 parameter DATA BITS = 2 + 7 + 1 + 1 + 1 + 3 + 64 + 6 + 102;
 2 //read data,
                      // {control packet, clause num, instruction}
      // outputs to AIS
                                                                                                                         parameter LO = 1'b0;
 5 aiq_export_info, //
                                                                                                                        parameter HI = 1'b1;
                                                                                                                         parameter X = 1'bx;
 6 aiq_pc_base,
 8 aiq_last_in_thread, //
      aig thread type, //
                                                                                                                   10
11
      aiq_valid_bits,
                                                                                                                  11
                                                                                                                  12
12
      aig thread id,
      aiq_instr,
                                                                                                                   13
                                                                                                                                                 write_rts;
14
                                                                                                                  14
                                                                                                                                                  write_rtr;
15
                                                                                                                  15
17 );
                                                                                                                  17
                                                                                                                         input [`SQ_VTX_CTL_PKT_WIDTH-1:0]
                                                                                                                                                                    aif ctl pkt q
                                                                                                                  18
                                                                                                                         input [1:0]
18
                                                                                                                                               aif export info;
                                                                                                                         input [6:0]
                                                                                                                                                 aif_pc_base_q;
20
                                                                                                                  20
                                                                                                                        input [5:0]
                                                                                                                                                 aif_thread_id_q;
21 parameter NUM_WORDS = 4;
                                                                                                                  21
                                                                                                                         input [0:0]
                                                                                                                                                  aif_last_in_group_q;
22 parameter ADDR_BITS = 2;
                                                                                                                  22
                                                                                                                         input [0:0]
                                                                                                                                                  aif_last_in_thread_q;
23
                                                                                                                  23
                                                                                                                         input [0:0]
                                                                                                                                                  aif_thread_type_q;
24 \hspace{1.5cm} \textit{//} \hspace{1.5cm} aiq\_export\_info + aiq\_pc\_base + aiq\_last\_in\_group
                                                                                                                  24
     /\!/ + aiq\_last\_in\_thread + aiq\_thread\_type + aiq\_context + aiq\_valid\_bits + aiq\_thread\_id + aiq\_instr
                                                                                                                                                  read_rts;
                                                                                                                         output
                                          Page 3 of 12
                                                                                                                                                            Page 4 of 12
                                                            Ex. 2095 - sq_alu_instr_queue.v
                                                                                                                                                                               Ex. 2095 - sq_alu_instr_queue.v
```

ATI 2095 LG v. ATI IPR2015-00326

```
wire [DATA BITS-1:0] write data;
 1
        input
                                    read rtr
                                                                                                                                           wire [DATA_BITS-1:0] read_data;
 2
 3
        output [1:0]
                            aiq export info;
        output [6:0]
                            aiq_pc_base;
        output [0:0]
                             aiq_last_in_group;
        output [0:0]
                             aiq_last_in_thread;
                                                                                                                                           // -- combinational logic --
        output [0:0]
                            aiq_thread_type;
        output [5:0]
        output [63:0]
                            aiq_valid_bits;
                                                                                                                                           wire [63:0] aif_valid_bits = aif_ctl_pkt_q[73:10];
                                                                                                                                           wire [6:0] gpr_base = aif_ctl_pkt_q[9:3];
        output [2:0]
                             aiq_context_id;
11
        output [101:0]
                                                                                                                                   11
                                                                                                                                           wire [2:0] aif_context_id = aif_ctl_pkt_q[2:0];
                                    aiq_instr;
12
13
                                    clk;
                                                                                                                                          wire [0:0] src_a_type = aif_instr_q[95];
        input
14
                                                                                                                                           wire [0:0] src_b_type = aif_instr_q[94];
        input
                                    reset;
15
                                                                                                                                   15
                                                                                                                                           wire [0:0] src c type = aif instr q[93];
16
                                                                                                                                   16
17
                                                                                                                                   17
                                                                                                                                           wire [2:0] src a sel = {aif instr q[95], aif instr q[87:86]};
        // -- internal signals --
                                                                                                                                           wire [2:0] src b sel = {aif instr q[94], aif instr q[79:78]};
18
                                                                                                                                   18
19
                                                                                                                                   19
                                                                                                                                           wire [2:0] src_c_sel = {aif_instr_q[93], aif_instr_q[71:70]};
20
                                                                                                                                   20
21
        wire [ADDR_BITS-1:0] write_ptr_q;
                                                                                                                                  21
                                                                                                                                           wire [4:0] vector_opcode = aif_instr_q[92:88];
22
                            write_en;
                                                                                                                                   22
23
                                                                                                                                   23
                                                                                                                                           wire [7:0] src_a_offset = aif_instr_q[87:80];
24
        wire [ADDR_BITS-1:0] read_ptr_q;
                                                                                                                                   24
                                                                                                                                           wire [7:0] src_b_offset = aif_instr_q[79:72];
25
                                                                                                                                           wire [7:0] src_c_offset = aif_instr_q[71:64];
                                                Page 5 of 12
                                                                                                                                                                                   Page 6 of 12
                                                                      Ex. 2095 - sq_alu_instr_queue.v
                                                                                                                                                                                                         Ex. 2095 - sq_alu_instr_queue.v
                                                                                                                                           wire [7:0] src_a_addr = src_a_type ? src_a_offset[5:0] + gpr_base : src_a_offset;
 2
        wire [5:0] scalar dest offset = aif instr q[13:8];
                                                                                                                                           wire [7:0] src b addr = src b type ? src b offset[5:0] + gpr base : src b offset;
        wire [5:0] vector dest offset = aif instr q[5:0];
                                                                                                                                           wire~[7:0] \quad src\_c\_addr = src\_c\_type~?~src\_c\_offset[5:0] + gpr\_base: src\_c\_offset;
 4
        wire [0:0] scalar_dest_addr_mode = aif_instr_q[14];
                                                                                                                                           wire [6:0] scalar_dest_addr = scalar_export ? {LO, scalar_dest_offset} : scalar_dest_offset
        wire [0:0] vector_dest_addr_mode = aif_instr_q[6];
                                                                                                                                          \label{eq:wire_final} \begin{tabular}{lll} wire [6:0] & vector\_dest\_addr &=& vector\_export &? & \{LO, & vector\_dest\_offset\} \\ vector\_dest\_offset + gpr\_base; & \end{tabular}
        wire [1:0] pred_sel = aif_instr_q[60:59];
        wire [0:0] scalar_dest_pred_sel = aif_instr_q[15];
                                                                                                                                          // - substitute actual gpr addresses into src and dest instruction fields
10
        wire [0:0] vector_dest_pred_sel = aif_instr_q[7];
                                                                                                                                           // - concatenate AIF input data and gpr addresses into fifo write data
11
                                                                                                                                   12
12
        // --> old emulator export decode
                                                                                                                                   13
                                                                                                                                           wire [101:0] instruction =
13
        //wire [0:0] scalar_export = ~pred_sel[1] & scalar_dest_pred_sel;
                                                                                                                                   14
14
        //wire [0:0] vector_export = ~pred_sel[1] & vector_dest_pred_sel;
                                                                                                                                   15
                                                                                                                                                 src a sel, src b sel, src c sel,
15
                                                                                                                                                vector opcode, src a addr, src b addr, src c addr,
16
        // --> new emulator export decode
                                                                                                                                                aif instr q[63:61], pred sel, aif instr q[58:16],
                                                                                                                                   17
17
        wire [0:0] scalar_export = aif_instr_q[15];
                                                                                                                                              scalar dest pred sel, scalar_dest_addr, vector_dest_pred_sel, vector_dest_addr
                                                                                                                                   18
18
        wire [0:0] vector export = aif instr q[15];
                                                                                                                                   19
19
                                                                                                                                   20
20
                                                                                                                                   21
                                                                                                                                           assign write_data = {aif_export_info, aif_pc_base_q,
21
       // - calculate gpr read addresses (src) and gpr write addresses (dst)
                                                                                                                                   22
                                                                                                                                                        aif\_last\_in\_group\_q, aif\_last\_in\_thread\_q, aif\_thread\_type\_q,
22
       // - if type is "constant", just pass the C number
                                                                                                                                   23
                                                                                                                                                        aif_context_id, aif_valid_bits, aif_thread_id_q,
23
        // - if type is "register", add the gpr base
                                                                                                                                   24
                                                                                                                                                        instruction);
24
        // - if exporting, do not add gpr base since dest is in this case an export address
25
                                                                                                                                           // - connect fifo read data to individual AIQ outputs
                                                Page 7 of 12
                                                                                                                                                                                   Page 8 of 12
                                                                      Ex. 2095 - sq_alu_instr_queue.v
                                                                                                                                                                                                         Ex. 2095 - sq alu instr queue.v
```

```
// fifo control for based fifo
2
      assign aiq_export_info = read_data[DATA_BITS-1:DATA_BITS-2];
                                                                                                                    2
      assign\ aiq\_pc\_base \quad = read\_data[DATA\_BITS-1-2:DATA\_BITS-2-7];
                                                                                                                          fifo regs ctl
     assign aiq_last_in_group = read_data[DATA_BITS-1-2-7];
      assign aiq_last_in_thread = read_data[DATA_BITS-1-2-7-1];
                                                                                                                            NUM_WORDS, ADDR_BITS
      assign aiq_thread_type = read_data[DATA_BITS-1-2-7-1-1];
       assign\ aiq\_context\_id\ = read\_data[DATA\_BITS-1-2-7-1-1-1:DATA\_BITS-2-7-1-1-1-3];
                                                                                                                           u_fifo_regs_ctl
     assign_aiq_valid_bits = read_data[DATA_BITS-1-2-7-1-1-1-3:DATA_BITS-2-7-1-1-1-3-64];
     assign aiq_thread_id = read_data[DATA_BITS-1-2-7-1-1-1-3-64:DATA_BITS-2-7-1-1-1-3-64-6];
                                                                                                                            .write_rts(write_rts),
                                                                                                                            .write_rtr(write_rtr),
                                                                                                                   11
                                                                                                                            .write_ptr_q(write_ptr_q),
12
      assign aiq_instr = read_data[101:0];
13 /*
                                                                                                                            .write_en(write_en),
14
       assign aiq_pc_base = read_data[181:175];
                                                                                                                            .read_rts_q(read_rts),
15
       assign aiq_last_in_group = read_data[174];
                                                                                                                            .read_rtr(read_rtr), // in
                                                                                                                    15
16
      assign aiq_last_in_thread = read_data[173];
                                                                                                                    16
17
      assign aiq_thread_type = read_data[172];
                                                                                                                            .read_ptr_q(read_ptr_q),
                                                                                                                    17
18
      assign aiq_context_id = read_data[171:169];
                                                                                                                    18
                                                                                                                            //.used_slots(used_count),
19
       assign aiq_valid_bits = read_data[168:105];
20
       assign aiq_thread_id = read_data[104:102];
                                                                                                                   19
                                                                                                                   20
21
      assign aiq_instr = read_data[101:0];
                                                                                                                            .clk(clk).
                                                                                                                   21
22 */
                                                                                                                            .reset(reset)
23
                                                                                                                   22
24
                                                                                                                   23
                                                                                                                   24
                                                                                                                           // fifo storage
                                           Page 9 of 12
                                                                                                                                                              Page 10 of 12
                                                             Ex. 2095 - sq_alu_instr_queue.v
                                                                                                                                                                                 Ex. 2095 - sq_alu_instr_queue.v
                                                                                                                         endmodule
2
      fifo regs
        ADDR_BITS, DATA_BITS, NUM_WORDS
4
5
       u_fifo_regs
7
8
         .wr_addr(write_ptr_q),
         .wr_en(write_en),
10
         .wr_data(write_data),
11
12
         .rd_addr(read_ptr_q),
13
         .rd_data(read_data),
14
15
16
17
18
19
20
21
      // -- one-bit state machines --
22
23
24
25
                                          Page 11 of 12
                                                                                                                                                              Page 12 of 12
                                                                                                                                                                                 Ex. 2095 - sq_alu_instr_queue.v
                                                             Ex. 2095 - sq_alu_instr_queue.v
```

```
1 'include "header.v"
                                                                                                             2 //-----
                                                                                                             3 'include "sq_defs.v"
 4 \hspace{0.5cm} // \hspace{0.1cm} \$ Id: // depot/r 400/ devel/parts\_lib/src/gfx/sq/ais/sq\_alu\_instr\_seq.v \# 13 \hspace{0.1cm} \$
 5 //
                                                                                                            5 module sq alu instr seq
 6 // $Change: 44201 $
 7 //
                                                                                                            7 alu_strap, // for alu0 vs alu1
 8 // Copyright: Trade secret of ATI Technologies, Inc.
                 © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
10 //
                                                                                                            10 aiq_rts, // rts from AIQ FIFO
                                                                                                            11 aiq_export_info, // {exp_id, pulse_sx}
11 //
               All rights reserved. This notice is intended as a precaution against
              inadvertent publication and does not imply publication or any waiver
                                                                                                            12 aiq_last_in_group, // last instruction in group (but not usually the last in the shader program)
                                                                                                            13 aiq_last_in_thread, // last instruction in shader program
13 //
              of confidentiality. The year included in the foregoing notice is the
14 //
                                                                                                            14 aiq_context_id,
              year of creation of the work.
                                                                                                                                        // context id
15 //
                                                                                                            15
                                                                                                                 aiq_thread_type, //
                                                                                                            16
                                                                                                                  aiq_thread_id, //
17
                                                                                                                  aiq_instr,
                                                                                                                                  // instruction
18 // sq_alu_instr_seq.v
                                                                                                            18
                                                                                                                  ais_rtr, // AIQ FIFO pop
19 //
                                                                                                            19
20 // - receives instruction from alu instr queue (AIQ)
                                                                                                            20
                                                                                                            21 // phase inputs
21 // - sends instruction to SP on the correct phase
                                                                                                            22
22 // - sends instruction to SP over four cycles
                                                                                                                  gpr_phase,
23 //
                                                                                                            23
                                                                                                                  alu_phase,
24 // issues:
                                                                                                            24
25 //-
                                                                                                                 // interface back to thread buffers
                                        Page 1 of 18
                                                                                                                                                    Page 2 of 18
                                                                                                                                                                        Ex. 2096 - sq alu instr seq.v
                                                           Ex. 2096 - sq alu instr seq.v
 1 ais_done,
                                                                                                             1 parameter HI = 1'b1;
 2 ais thread type q,
                                                                                                                 parameter X = 1'bx;
     ais_thread_id_q,
5 // interface back to SX via exp alloc
 6 ais_free_done, // export buffer dealloc to SX
                          // export buffer id
 9 // to ais output
                                                                                                                  input alu strap;
                                                                                                            10
11
     ais instr stall,
                                                                                                            11
                                                                                                            12
                                                                                                                 input [1:0] aig export info;// {exp id, pulse sx}
12 //ais ld isr,
13 ais_acs_rd_rts, // alu constant store read address valid
                                                                                                            13 input [0:0] aiq_last_in_group; // last instruction flag
14 ais_acs_rd_addr, // alu constant store read address (to ais_output)
                                                                                                            14 input [0:0] aiq_last_in_thread; // last instruction flag
15
                                                                                                            15
                                                                                                                 input [0:0] aiq thread type;// 0: pixel, 1: vertex
16
      //aluconst_context_valid,
                                                                                                                  input [2:0] aiq_context_id; // context_id (from ctl packet)
17
                                                                                                            17
                                                                                                                  input [5:0] aiq_thread_id; // clause number
                                                                                                                 input [101:0] aiq instr; // instruction
18 busy,
                                                                                                            18
19 clk,
20 reset
                                                                                                            20 output [0:0] ais_rtr;
                                                                                                            21
21 ):
                                                                                                                  reg [0:0] ais rtr;
                                                                                                            22
23 // -- parameters --
                                                                                                            23
                                                                                                                 input [1:0] gpr_phase; //
24
25 parameter LO = 1'b0;
                                                                                                                 input [0:0] alu_phase; //
                                        Page 3 of 18
                                                                                                                                                   Page 4 of 18
                                                           Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                       Ex. 2096 - sq_alu_instr_seq.v
```

ATI 2096 LG v. ATI IPR2015-00326

```
2
                                                                                                                                      //input [7:0] aluconst_context_valid;
       output [0:0] ais_acs_rd_rts;// alu constant store read address (logical addr - up to 256 consts)
       reg [0:0] ais_acs_rd_rts; // alu constant store read address (logical addr - up to 256 consts)
                                                                                                                                       output busy;
                                                                                                                                       input clk;
       output [0:0] ais_free_done; // export buffer dealloc to SX
       output [0:0] ais_free_id_q;
       output [0:0] ais_thread_type_q;
11
       output [5:0] ais_thread_id_q;
                                                                                                                                       // -- internal signals --
12
       output [0:0] ais_done;
13
       reg [0:0] ais_thread_type_q;
14
       reg [5:0] ais_thread_id_q;
                                                                                                                                      reg [2:0] acs_current_state;
15
                                                                                                                               15
       reg [0:0] ais done;
                                                                                                                                      reg [2:0] ais current state;
                                                                                                                               16
16
       reg [0:0] ais free done;
17
                                                                                                                              17
                                                                                                                                      reg ais start;
18
                                                                                                                               18
19
       output [0:0] ais instr start;
                                                                                                                               19
                                                                                                                                      wire ca fetch = ~aiq instr[101];
                                                                                                                                      wire cb_fetch = ~aiq_instr[98];
20
       output [0:0] ais instr stall;
                                                                                                                               20
                                                                                                                                      wire cc_fetch = ~aiq_instr[95];
                                                                                                                              21
21
22
      output [8:0] ais_acs_rd_addr;
consts)
                                       // alu constant store read address (logical addr - up to 512
23
                                                                                                                              23
                                                                                                                                      wire [8:0] ca_addr = {1'b0, aiq_instr[87:80]};
24
       reg [0:0] ais_instr_start;
                                                                                                                              24
                                                                                                                                       wire [8:0] cb_addr = {1'b0, aiq_instr[79:72]};
25
                                                                                                                                       wire [8:0] cc_addr = {1'b0, aiq_instr[71:64]};
       reg [8:0] ais_acs_rd_addr; // alu constant store read address (logical addr - up to 512 consts)
                                              Page 5 of 18
                                                                                                                                                                              Page 6 of 18
                                                                      Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                                     Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                      // -- Input Staging Register --
2
       //reg [2:0] ais state q;
                                                                                                                                      // - need to send the vector type and the thread id back to the thread buffers when
                                                                                                                                      // the all the instructions we wanted to run for this thread are done (this will
       // -- module instatiations --
                                                                                                                                      // cause the thread to become valid again)
                                                                                                                                      // - register this info in from the AIS on an AIQ pop in order to hold it until the
                                                                                                                                      // AIS is done
                                                                                                                                      reg [0:0] ais_last_in_group_q;
10
       // -- combinational logic --
                                                                                                                               10
                                                                                                                                      reg [0:0] ais_last_in_thread_q;
11
                                                                                                                              11
                                                                                                                                      reg [0:0] ais_free_id_q;
12
                                                                                                                               12
                                                                                                                                       reg [0:0] ais_pulse_sx_q;
13
       wire busy = (|acs_current_state) | (|ais_current_state);
                                                                                                                               13
14
                                                                                                                                       always @(posedge clk)
15
       // - constant store valid (one bit per context from ALU Const Store)
                                                                                                                               15
                                                                                                                                       begin
16
                                                                                                                                            if ( ais_rtr )
17
       //wire [0:0] aluconst_valid;
18
                                                                                                                                               ais_thread_type_q <= aiq_thread_type;
19
       //assign aluconst_valid = aluconst_context_valid[aiq_context_id];
                                                                                                                                              ais_thread_id_q <= aiq_thread_id;
                                                                                                                                              //ais_context_id_q <= aiq_context_id;
20
                                                                                                                               20
21
                                                                                                                                              ais last in group q <= aiq last in group;
22
       // -- registers --
                                                                                                                                              ais last in thread q <= aiq last in thread;
23
                                                                                                                               23
                                                                                                                                              ais\_free\_id\_q \mathrel{<=} aiq\_export\_info[1];
24
                                                                                                                               24
                                                                                                                                               ais_pulse_sx_q <= aiq_export_info[0];
                                                                                                                               25
                                              Page 7 of 18
                                                                                                                                                                              Page 8 of 18
                                                                      Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                                     Ex. 2096 - sq alu instr seq.v
```

```
// ACS (ALU Constant Store) Read State Machine:
             else
 2
              begin
                                                                                                                                     // - assert the ACS read rts
               ais\_thread\_type\_q \mathrel{<=} ais\_thread\_type\_q;
                                                                                                                                     // - select the ACS read address from the 3 source addresses present in the instruction
               ais\_thread\_id\_q \le ais\_thread\_id\_q;
                                                                                                                                     // - kick off the AIS stage
               //ais_context_id_q <= ais_context_id_q;
               ais\_last\_in\_group\_q \mathrel{<=} ais\_last\_in\_group\_q;
                                                                                                                                     // AIS (ALU Instruction Sequencer) State Machine:
                ais\_last\_in\_thread\_q \mathrel{<=} ais\_last\_in\_thread\_q;
                                                                                                                                   /\!/ - load the ISR in the ais_output module (to hold dest write addrs after AIQ has been popped)
                ais\_free\_id\_q \mathrel{<=} ais\_free\_id\_q;
                ais_pulse_sx_q <= ais_pulse_sx_q;
                                                                                                                                     // - pop the AIQ
10
11
                                                                                                                                    // ACS read state machine
12
                                                                                                                             13
13
                                                                                                                                    parameter ACS0 = 3'b000;
                                                                                                                              14
14
                                                                                                                              15
                                                                                                                                     parameter ACS1 = 3'b001:
15
       // -- one-bit state machines --
                                                                                                                              16
                                                                                                                                     parameter ACS2 = 3'b010:
16
                                                                                                                                     parameter ACS3 = 3'b011:
17
                                                                                                                                     parameter ACS4 = 3'b100:
18
                                                                                                                                     parameter ACS5 = 3'b101;
19
                                                                                                                                     parameter ACS6 = 3'b110;
20
       // -- state machines --
                                                                                                                                     parameter ACS7 = 3'b111;
21
                                                                                                                             22
22
                                                                                                                             23
                                                                                                                                     reg [2:0] acs_next_state;
23
       // - the following two state machines work together in a staged manner (1st triggers 2nd)
                                                                                                                             24
24
       // - each cycles thru 8 states to match the 8 cycle transfer to the SP
                                                                                                                             25
                                                                                                                                     // state and output registers
25
                                                                                                                                      always @(posedge clk)
                                               Page 9 of 18
                                                                                                                                                                            Page 10 of 18
                                                                     Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                                   Ex. 2096 - sq_alu_instr_seq.v
         begin
                                                                                                                               2
 2
             if (reset)
                                                                                                                                          case (acs current state)
              begin
                acs current state <= ACS0;
                                                                                                                                          // - start when AIQ output is valid and gpr read phase is srcB
 5
            end
                                                                                                                                             // (this phase is chosen due to ACS read latency - if latency changes, this also must
             else
                                                                                                                               7
                                                                                                                                         // - also need to start on the correct alu_phase (opposite of alu_strap since this is starting
                                                                                                                                            // 4 cycles early)
                acs_current_state <= acs_next_state;
                                                                                                                               9
            end
                                                                                                                                   if (aiq\_rts \ \& \ (gpr\_phase = `SQ\_SRCB\_PHASE) \ \& \ (alu\_phase = ~alu\_strap) \ ) // \ \& \ aluconst\_valid \ )
10
          end
11
                                                                                                                                               begin
12
       // next state logic
                                                                                                                                              if (ca_fetch)
                                                                                                                                                                       // read first constant
13
                                                                                                                                                  begin
14
            aiq_rts or gpr_phase or
                                                                                                                              15
                                                                                                                                                     ais acs rd rts = HI;
15
            ca_fetch or ca_addr or cb_fetch or cb_addr or cc_fetch or cc_addr or
                                                                                                                                                     ais acs rd addr = ca addr;
16
               acs_current_state
                                                                                                                                                    end
                                                                                                                              17
17
                                                                                                                              18
                                                                                                                                              acs_next_state = ACS1;
18
        begin
                                                                                                                              19
                                                                                                                                             end
19
            // default assignments
                                                                                                                              20
                                                                                                                                            end
20
            acs next state = ACS0;
                                                                                                                              21
21
                                                                                                                              22
                                                                                                                                            ACS1:
22
          ais start = LO:
                                                                                                                              23
                                                                                                                                             begin
23
          ais\_acs\_rd\_rts = LO; \qquad /\!/ \ read\ request\ valid\ to\ the\ alu\ const\ store
                                                                                                                              24
                                                                                                                                             if (cb_fetch)
24
          ais_rtr = LO:
                                                                                                                              25
25
            ais_acs_rd_addr = ca_addr;
                                                                                                                                                     ais_acs_rd_rts = HI;
                                                                                                                                                                           // read second constant
                                              Page 11 of 18
                                                                                                                                                                            Page 12 of 18
                                                                     Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                                   Ex. 2096 - sq_alu_instr_seq.v
```

```
ais_acs_rd_addr = cb_addr;
                                                                                                                               ACS6: begin acs_next_state = ACS7; end
2
                                                                                                                   2
                   end
             acs_next_state = ACS2;
                                                                                                                               ACS7:
              end
                                                                                                                             // - pop the AIQ
             ACS2:
                                                                                                                                ais_rtr = HI;
                                                                                                                              acs_next_state = ACS0;
             if (cc_fetch)
                   ais_acs_rd_rts = HI; // read third constant
                                                                                                                   10
                                                                                                                              endcase // case(acs_current_state)
11
                   ais_acs_rd_addr = cc_addr;
                                                                                                                  11
                                                                                                                          end // always @ (*)
12
                                                                                                                          // - end acs read state machine
13
               acs next state = ACS3;
14
15
                                                                                                                  15
                                                                                                                        // AIS state machine
16
                                                                                                                  16
             ACS3: begin acs next state = ACS4; end
17
                                                                                                                  17
                                                                                                                        parameter AIS0 = 3'b000:
18
            ACS4
                                                                                                                  18
                                                                                                                        parameter AIS1 = 3'b001;
19
           begin
                                                                                                                  19
                                                                                                                        parameter AIS2 = 3'b010;
20
             ais start = HI;
                                                                                                                  20
                                                                                                                         parameter AIS3 = 3'b011;
21
                                                                                                                  21 parameter AIS4 = 3'b100;
             acs_next_state = ACS5;
22
                                                                                                                  22
                                                                                                                        parameter AIS5 = 3'b101;
23
                                                                                                                  23
                                                                                                                        parameter AIS6 = 3'b110;
24
             ACS5: begin acs_next_state = ACS6; end
                                                                                                                  24
                                                                                                                         parameter AIS7 = 3'b111;
25
                                                                                                                   25
                                         Page 13 of 18
                                                                                                                                                            Page 14 of 18
                                                               Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                  Ex. 2096 - sq_alu_instr_seq.v
      //reg [2:0] ais_current_state;
                                                                                                                            ais_instr_start = LO;
2
      reg [2:0] ais_next_state;
                                                                                                                            ais instr stall = LO;
                                                                                                                            //ais_ld_isr = LO;
4
      // state and output registers
                                                                                                                            ais done = LO:
5
      always @(posedge clk)
                                                                                                                            ais_free_done = LO;
        begin
7
           if (reset)
                                                                                                                              case (ais_current_state)
            begin
                                                                                                                               AIS0:
              ais_current_state <= AIS0;
10
           end
                                                                                                                   10
                                                                                                                                 // - wait until this machine is started by the AIQ read SM
11
                                                                                                                  11
                                                                                                                               ais\_instr\_stall = HI;
12
                                                                                                                  12
                                                                                                                                if (ais_start)
13
              ais_current_state <= ais_next_state;
14
                                                                                                                                   ais_instr_start = HI;
15
                                                                                                                                ais_instr_stall = LO;
16
                                                                                                                                 ais_next_state = AIS1;
17
     // next state logic
18
      always @(
19
           ais start or
20
                                                                                                                   20
             ais current state
                                                                                                                               AIS1: begin ais_next_state = AIS2; end
21
                                                                                                                  21
22
                                                                                                                  22
                                                                                                                                AIS2: begin ais next state = AIS3; end
23
          // default assignments
                                                                                                                  23
24
            ais_next_state = AIS0;
                                                                                                                  24
                                                                                                                                AIS3: begin ais_next_state = AIS4; end
25
                                                                                                                   25
                                          Page 15 of 18
                                                                                                                                                            Page 16 of 18
                                                               Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                 Ex. 2096 - sq_alu_instr_seq.v
```

```
1
               AIS4: begin ais_next_state = AIS5; end
                                                                                                                                        1
 2
                                                                                                                                       2 endmodule
               AIS5: begin ais_next_state = AIS6; end
               AIS6: begin ais_next_state = AIS7; end
               AIS7:
            // - load ISR in ais_output
             /\!/ \text{ - send ais\_done back to the thread buffers when this is the last instruction in a group of}
10
    alu instructions
                                                                                                                                       10
              // AND it is not the LAST group of alu instructions (since thread is no longer in the
11
12
                                                                                                                                      11
      reservation station buffer)
13
14
      /\!/ - send free_done when pulse_sx is set, or this is the last instruction of a pixel shader (since this
                                                                                                                                      12
                                                                                                                                       13
15
              // is when the pixel export is done)
                                                                                                                                      14
16
            begin
                                                                                                                                      15
               //ais_ld_isr = HI;
17
                                                                                                                                      16
18
              ais_next_state = AIS0;
                                                                                                                                      17
              if (\ ais\_last\_in\_group\_q \ \& \ {\sim} ais\_last\_in\_thread\_q \ ) \ ais\_done = HI;
20
              if \ (\ ais\_pulse\_sx\_q \ | \ (ais\_last\_in\_thread\_q \ \& \ \sim ais\_thread\_type\_q) \ ) \ ais\_free\_done = HI;
22
23
              endcase // case(ais_current_state)
24
         end // always @ (*)
25
         // - end ais state machine
26
                                                 Page 17 of 18
                                                                                                                                                                                       Page 18 of 18
                                                                          Ex. 2096 - sq_alu_instr_seq.v
                                                                                                                                                                                                                Ex. 2096 - sq_alu_instr_seq.v
```

```
1 'include "header.v"
                                                                                                                 2 //-----
 4 \hspace{0.5cm} /\hspace{0.1cm}/\hspace{0.1cm} \$ Id: /\hspace{-0.1cm}/\hspace{0.1cm} depot/r400/devel/parts\_lib/src/gfx/sq/ca/sq\_thread\_arb.v\#19~\$
                                                                                                                 4 `include "../misc/sq_defs.v"
 5 //
 6 // $Change: 43237 $
                                                                                                                 6 module sq_thread_arb
 7 //
 8 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                                8 arb_type_strap, // tex = 1, alu = 0
                 © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
                                                                                                                       state_read_phase, // share read access between tex and alu arbs
10 //
                                                                                                                10
11 //
                All rights reserved. This notice is intended as a precaution against
                                                                                                                11
                                                                                                                     // vertex and pixel thread buffer interface
               inadvertent publication and does not imply publication or any waiver
                                                                                                                12
13 //
                of confidentiality. The year included in the foregoing notice is the
                                                                                                                13 vtx_req_q, // 16 vtx_thread_buff requests
14 //
               year of creation of the work.
                                                                                                                14
15 //
                                                                                                                15
                                                                                                                      vtx_winner_q, // winning vertex thread_id sent back to Vertex Thread Buffer
                                                                                                                16
                                                                                                                       vtx\_winner\_ack, \qquad /\!/ \ request \ acknowledge \ - \ indicates \ to \ TB \ that \ the \ winner \ is \ valid
17
                                                                                                                18
18 // sq thread arb.v
                                                                                                                     vtx state,
19 //
                                                                                                                19
                                                                                                                      vtx_status,
20 // - pick a thread from requests being sent by the Vertex and Pixel Thread Buffers
                                                                                                                20
21 // - thread state and status is selected by the winner in each thread buffer and sent
                                                                                                                21 pix_req_q, // 16 pix_thread_buff requests
                                                                                                                22
22 // back to the arbiter for the final muxing (btwn vtx and pix) to the CFS
23 //
                                                                                                                23
                                                                                                                      pix_winner_q, // winning pixel thread_id sent back to Pixel Thread Buffer
24 // issues:
                                                                                                                24
                                                                                                                      pix_winner_ack, //
25 //-
                                         Page 1 of 28
                                                                                                                                                         Page 2 of 28
                                                                Ex. 2097 - sq thread arb.v
                                                                                                                                                                                Ex. 2097 - sq thread arb.v
 1 pix_state,
                       //
                                                                                                                 1 parameter LO = 1'b0;
                                                                                                                     parameter HI = 1'b1;
      pix_status,
                                                                                                                       parameter X = 1'bx;
      // control flow sequencer interface
 6 arb_rts0, // ready to send the winner to CFS0
 7 arb_rts1, // ready to send the winner to CFS1
 8 arb state, // the state sent to the CFS
      arb status, // the status sent to the CFS
       arb_thread_type, // vtx or pix
                                                                                                                10
                                                                                                                       input [0:0] arb_type_strap;
11
                                                                                                                11
                                                                                                                      input [0:0] state_read_phase;
12 cfs_rtr0, // CFS0 can accept a thread
                                                                                                                12
     cfs_rtr1, // CFS1 can accept a thread (for alu cfs's)
                                                                                                                                            vtx_req_q;
14
                                                                                                                14
                                                                                                                      input ['SQ_VTX_STATE_WIDTH-1:0] vtx_state;
15 cfs1 enable, /\!/ enable sending packets to CFS1 (this a local register setting: 16 SQ_FLOW_CTLONE_ALU)
                                                                                                                15
                                                                                                                       input ['SQ_VTX_STATUS_WIDTH-1:0] vtx_status;
17
                                                                                                                17
                                                                                                                       output [3:0]
                                                                                                                                             vtx winner q;
18
                                                                                                                18
                                                                                                                      output [0:0]
                                                                                                                                            vtx winner ack;
19
     reset
                                                                                                                20
                                                                                                                     input [15:0]
                                                                                                                                            pix_req_q;
21
                                                                                                                      input ['SQ_PIX_STATE_WIDTH-1:0] pix_state;
                                                                                                                21
22
     // -- parameters --
                                                                                                                22
                                                                                                                       input ['SQ_PIX_STATUS_WIDTH-1:0] pix_status;
                                                                                                                23
24
      parameter STATE_WIDTH = 8; //
                                                                                                                24 output [3:0]
                                                                                                                                              pix_winner_q;
     parameter STATUS WIDTH = 8; //
                                                                                                                                              pix_winner_ack;
                                                                                                                     output [0:0]
                                                                                                                                                        Page 4 of 28
                                         Page 3 of 28
                                                                Ex. 2097 - sq thread arb.v
                                                                                                                                                                                Ex. 2097 - sq thread arb.v
```

ATI 2097 LG v. ATI IPR2015-00326

```
2
     // interface to the control flow sequencer
                                                                                                        reg [STATE WIDTH-1:0]
                                                                                                                                   arb state;
      output [0:0]
                               arb rts0;
                                                                                                        reg [STATUS WIDTH-1:0]
                                                                                                                                         arb status:
                                arb_rts1;
      output [STATE_WIDTH-1:0]
                                   arb_state;
                                                                                                        // one bit state machine
      output [STATUS_WIDTH-1:0]
                                     arb_status
                                arb_thread_type;
                                cfs_rtr0;
                                                                                                        // thread_type_sm
      input [0:0]
                                cfs_rtr1;
                                                                                                        reg [0:0] type_winner;
                                                                                                        reg [0:0] vtx_winner_ack;
11
      input [0:0]
                                cfs1_enable;
                                                                                                        reg [0:0] pix_winner_ack;
12
                                                                                                  12
13
      input clk;
                                                                                                        // thread read sm
14
                                                                                                        reg [0:0] ld_winner;
      input reset;
15
                                                                                                  15
                                                                                                        reg [0:0] arb_rts;
16
                                                                                                  16
17
                                                                                                  17
18
      // -- internal signals --
                                                                                                  18
19
                                                                                                  19
                                                                                                        // -- module instatiations --
20
                                                                                                  20
21
      reg [3:0] vtx_winner_q;
                                                                                                  21
22
      reg [3:0] pix_winner_q;
                                                                                                  22
23
                                                                                                  23
24
      reg [0:0] vtx_winner_vld_q;
                                                                                                  24
                                                                                                        // -- combinational logic --
25
      reg [0:0] pix_winner_vld_q;
                                    Page 5 of 28
                                                                                                                                       Page 6 of 28
                                                        Ex. 2097 - sq_thread_arb.v
                                                                                                                                                           Ex. 2097 - sq_thread_arb.v
                                                                                                            5'h17; end
2
     // * arbiter *
                                                                                                           5'h16; end
     // - the highest thread number has the highest priority, and verts have priority over pixels
      // - try with loop/tree coding style
                                                                                                           32'b0000_0000_001?_????_????_????_????: begin winner_vld = HI; winner =
                                                                                                       32'50000\_0000\_0001\_????\_????\_????\_???? begin winner_vld = HI; winner = 5'h14; end
     wire [31:0] req_vector = {vs_req, ps_req};
                                                                                                            32'b0000\_0000\_0000\_1???\_????\_????\_????: begin winner\_vld = HI; winner = 0.0000\_0000\_0000\_1???\_????
                                                                                                  10
      reg [0:0] winner_vld;
                                                                                                            32'b0000\_0000\_0000\_01??\_????\_????\_????: \ begin \ winner\_vld = HI; \ winner = 0.0000\_0000\_01??\_????
      reg [4:0] winner;
                                                                                                       5'h12: end
10
                                                                                                            5'h11; end
11
      always @(req_vector)
                                                                                                       32'b0000\_0000\_0000\_0001\_????\_????\_????: \ begin \ winner\_vld = HI; \ winner = 5'h10; \ end
12
13
                                                                                                            32'b0000_0000_0000_0000_1???_????_????: begin winner_vld = HI; winner =
14
15
         32'b1???\_????\_????\_????\_????\_????: begin winner\_vld = HI; winner = 5'h1f; \\
                                                                                                            5'h0e: end
16
17
         32'b01??_????_????_????_????_???? begin winner_vld = HI; winner = 5'h1e;
                                                                                                            32'b0000_0000_0000_0000_001?_????_????: begin winner_vld = HI; winner =
                                                                                                       5'h0d; end
18
19
         32'b001?_????_????_????_????_????_????: begin winner_vld = HI; winner =
                                                                                                           32'b0000_0000_0000_0000_0001_????_????: begin winner_vld = HI; winner =
20
21
         32'b0001_????_????_????_????_????. begin winner_vld = HI; winner =
                                                                                                       32'b0000\_0000\_0000\_0000\_0000\_1???\_?????: begin winner_vld = HI; winner = 5'h0b; end
    32'b0000\_1???\_????\_????\_????\_???? begin winner_vld = HI; winner = 5'h1b; end
22
23
                                                                                                            32'b0000\_01??\_????\_????\_????\_????: begin winner_vld = HI; winner = 5'h1a; end
                                                                                                            26
27
         32'b0000_001?_????_????_????_????_????: begin winner_vld = HI; winner =
    5'h19: end
                                                                                                           5'h08; end
         32'b0000_0001_????_????_????_????_????: begin winner_vld = HI; winner =
    5'h18; end
                                    Page 7 of 28
                                                                                                                                       Page 8 of 28
                                                        Ex. 2097 - sq thread arb.v
                                                                                                                                                           Ex. 2097 - sq_thread_arb.v
```

```
32'b0000 0000 0000 0000 0000 1??? ????: begin winner vld = HI; winner =
                                                                                                                                                                                  begin
                                                                                                                                                                        2
                                                                                                                                                                                    casez (vtx req q)
        32'b0000\_0000\_0000\_0000\_0000\_0000\_01??\_????: begin winner_vld = HI; winner = 5'h06; end
                                                                                                                                                                                      16'b0000 0000 0000 0000: begin vtx winner vld = LO; vtx winner = 4'hf; end
        32'b0000\_0000\_0000\_0000\_0000\_0000\_001?\_???? begin winner_vld = HI; winner = 5'h05; end
                                                                                                                                                                                       16'b1000_0000_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'hf; end
                                                                                                                                                                                       16'b?100_0000_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'he; end
                32'b0000_0000_0000_0000_0000_0001_????: begin winner_vld = HI; winner =
                                                                                                                                                                                       16'b??10_0000_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'hd; end
                32'b0000\_0000\_0000\_0000\_0000\_0000\_0000\_1???: \ begin \ winner\_vld = HI; \ winner = 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 + 0.0000 +
                                                                                                                                                                                      16'b???1 0000 0000 0000: begin vtx winner vld = HI; vtx winner = 4'hc; end
10
        5'h03: end
                                                                                                                                                                                       16'b????_1000_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'hb; end
                16'b????_?100_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'ha; end
        5'h02: end
                                                                                                                                                                                       16'b????_??10_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'h9; end
                32'b0000_0000_0000_0000_0000_0000_001?: begin winner_vld = HI; winner =
        5'h01; end
                                                                                                                                                                                       16'b????_???1_0000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'h8; end
                32'b0000_0000_0000_0000_0000_0000_0001: begin winner_vld = HI; winner =
                                                                                                                                                                                       16'b????_????_1000_0000: begin vtx_winner_vld = HI; vtx_winner = 4'h7; end
                                                                                                                                                                                       16'b????_????_?100_0000: begin vtx_winner_vld = HI; vtx_winner = 4'h6; end
17
                5'h00: end // winner is really don't care here
                                                                                                                                                                                       16'b????_????_??10_0000: begin vtx_winner_vld = HI; vtx_winner = 4'h5; end
19
                default:
                                                     begin winner vld = X: winner = {5{X}}; end
                                                                                                                                                                                      16'b???? _???? _???1_0000: begin vtx_winner_vld = HI; vtx_winner = 4'h4; end
                                                                                                                                                                      15
20
                 endcase
                                                                                                                                                                                      16'b???? ???? ???? 1000: begin vtx winner vld = HI: vtx winner = 4'h3: end
                                                                                                                                                                      16
21
            end // always @ (req_vector)
                                                                                                                                                                      17
                                                                                                                                                                                      16'b???? ???? ???? ?100: begin vtx winner vld = HI: vtx winner = 4'h2: end
22
                                                                                                                                                                                      16'b???? _???? _???? _??10: begin vtx_winner_vld = HI; vtx_winner = 4'h1; end
                                                                                                                                                                      18
23
                                                                                                                                                                                       16'b???? ???? ???? ???1: begin vtx winner vld = HI: vtx winner = 4'h0: end
                                                                                                                                                                      19
24
          // - vertex request priority encoder
                                                                                                                                                                      20
                                                                                                                                                                                      default
                                                                                                                                                                                                             begin vtx winner vld = X; vtx winner = 4'bxxxx; end
25
                                                                                                                                                                      21
                                                                                                                                                                                       endcase
26
                    vtx_winner_vld;
                                                                                                                                                                      22
27
          reg [3:0] vtx_winner;
                                                                                                                                                                      23
28
                                                                                                                                                                      24
          always @(vtx_req_q)
                                                                                                                                                                                // - pixel request priority encoder
                                                              Page 9 of 28
                                                                                                                                                                                                                                   Page 10 of 28
                                                                                               Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                                                                                                     Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                      default:
                                                                                                                                                                                                             begin pix winner vld = LO; pix winner = 4'bxxxx; end
 2
         reg
                     pix winner vld;
                                                                                                                                                                                       endcase
          reg [3:0] pix winner;
                                                                                                                                                                                  end
 4
 5
          always @(pix req q)
           begin
                                                                                                                                                                                // - if cfs1 is enabled alternate btwn rts0 and rts1
              casez (pix_req_q)
                                                                                                                                                                                // - if cfs1 is disabled, mask rts1 and always use rts0
                //16'b0000 0000 0000 0000: begin pix winner vld = LO; pix winner = 4'hf; end
                                                                                                                                                                                 // - what is the algorithm here? really want to send the thread to the CFS that's available
                16'b1000_0000_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'hf; end
                                                                                                                                                                      10
                                                                                                                                                                                // to cfs0 if both are available)
10
                16"b?100\_0000\_0000\_0000: begin\ pix\_winner\_vld = HI;\ pix\_winner = 4"he;\ end
                                                                                                                                                                                // - so getting rid of forced toggle btwn cfs0 and cfs1 - remember to to comment out cfs_turn
11
                16'b??10_0000_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'hd; end
                                                                                                                                                                      12
12
                16'b???1_0000_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'hc; end
                                                                                                                                                                                //assign arb_rts0 = arb_rts & (~cfs_turn | ~cfsl_enable);
13
                16'b????_1000_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'hb; end
                                                                                                                                                                                //assign arb_rts1 = arb_rts & cfs_turn & cfs1_enable;
14
                16'b????_?100_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'ha; end
                                                                                                                                                                      15
15
                16'b????_??10_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'h9; end
                                                                                                                                                                                //wire [0:0] cfs_rtr = cfs_rtr0 | cfs_rtr1;
                16'b????_???1_0000_0000: begin pix_winner_vld = HI; pix_winner = 4'h8; end
                                                                                                                                                                      17
17
                16'b????_????_1000_0000: begin pix_winner_vld = HI; pix_winner = 4'h7; end
                                                                                                                                                                      18
                                                                                                                                                                                wire [0:0] send_to_cfs0 = cfs_rtr0;
18
                16'b????_????_?100_0000: begin pix_winner_vld = HI; pix_winner = 4'h6; end
                                                                                                                                                                      19
                                                                                                                                                                                wire [0:0] send to cfs1 = ~cfs rtr0 & cfs rtr1 & cfs1 enable;
19
                16'b????_????_??10_0000: begin pix_winner_vld = HI; pix_winner = 4'h5; end
                                                                                                                                                                      20
                16'b????_????_???1_0000: begin pix_winner_vld = HI; pix_winner = 4'h4; end
20
                                                                                                                                                                                assign arb rts0 = arb rts & send to cfs0:
                                                                                                                                                                      21
21
                16'b????_????_1000: begin pix_winner_vld = HI; pix_winner = 4'h3; end
                                                                                                                                                                      22
                                                                                                                                                                                assign arb_rts1 = arb_rts & send_to_cfs1;
22
                16'b???? ???? ???? ?100: begin pix winner vld = HI: pix winner = 4'h2: end
                                                                                                                                                                      23
23
                16'b???? ???? ????? ??10: begin pix winner vld = HI; pix winner = 4'h1; end
                                                                                                                                                                      24
                                                                                                                                                                                wire [0:0] arb xfc0 = arb rts0 & cfs rtr0;
24
                16'b???? ???? ???? ???! begin pix winner vld = HI: pix winner = 4'h0: end
                                                                                                                                                                      25
                                                                                                                                                                                wire [0:0] arb xfc1 = arb rts1 & cfs rtr1;
25
                //default:
                                        begin pix winner vld = X; pix winner = 4'bxxxx; end
                                                             Page 11 of 28
                                                                                                                                                                                                                                   Page 12 of 28
                                                                                               Ex. 2097 - sq thread arb.v
                                                                                                                                                                                                                                                                     Ex. 2097 - sq thread arb.v
```

```
wire [0:0] arb xfc = arb xfc0 | arb xfc1;
                                                                                                                                       case (type_winner)
 2
                                                                                                                                        HI: arb_status = vtx_status;
                                                                                                                                        LO: arb_status = pix_status;
                                                                                                                                        /\!/default: arb\_status = \{STATUS\_WIDTH\{X\}\};
       // -- Arb Output Mux --
       // - choose between tex state/status and pix state/status depending on overall winner
       /\!/ - pix \ tex \ does \ have \ LOD \ (PIX\_CTL\_PKT\_WIDTH \ and \ CTL\_PKT\_WIDTH \ have \ lod)
                                                                                                                                      // -- registers --
11
       // - pix alu has no lod
                                                                                                                              11
12
                                                                                                                             12
13
                                                                                                                             13
                                                                                                                                     // register the winner based on ld_winner
       always @(type winner or vtx state or pix state)
                                                                                                                             14
14
15
         //arb_state = {STATE_WIDTH{LO}};
                                                                                                                             15
                                                                                                                                     always @(posedge clk)
16
                                                                                                                             16
         case (type winner)
17
          HI: arb_state = vtx_state; // these are unequal - msb's get 0's by above assignment
                                                                                                                             17
                                                                                                                                           if (reset)
18
          LO: arb state = pix state;
                                                                                                                              18
                                                                                                                                            begin
19
          //default: arb state = {STATE WIDTH{X}};
                                                                                                                              19
                                                                                                                                             vtx winner q <= 4'h0;
20
         endcase
                                                                                                                              20
                                                                                                                                             vtx\_winner\_vld\_q \mathrel{<\!\!=} LO;
21
        end
                                                                                                                             21
                                                                                                                                            pix_winner_q <= 4'h0;
22
                                                                                                                              22
                                                                                                                                            pix\_winner\_vld\_q \mathrel{<=} LO;
23
       always @(type_winner or vtx_status or pix_status)
                                                                                                                             23
24
                                                                                                                             24
                                                                                                                                           else if (ld_winner)
25
         //arb_status = {STATUS_WIDTH{LO}};
                                                                                                                              25
                                              Page 13 of 28
                                                                                                                                                                            Page 14 of 28
                                                                        Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                                      Ex. 2097 - sq_thread_arb.v
             vtx winner q <= vtx winner;
                                                                                                                                      always @(posedge clk)
 2
                                                                                                                               2
               vtx_winner_vld_q <= vtx_winner_vld;
                                                                                                                                      begin
                                                                                                                                           if ( reset ) cfs turn <= LO;
             pix_winner_q <= pix_winner;
               pix\_winner\_vld\_q \mathrel{<=} pix\_winner\_vld;
              end
                                                                                                                                            case ( toggle turn )
             else
                                                                                                                                              cfs turn <= ~cfs turn;
               vtx_winner_q <= vtx_winner_q;
               vtx\_winner\_vld\_q \mathrel{<=} vtx\_winner\_vld\_q;
                                                                                                                                              cfs_turn <= cfs_turn;
                pix\_winner\_q \mathrel{<=} pix\_winner\_q;
10
                                                                                                                              10
11
                pix\_winner\_vld\_q \mathrel{<=} pix\_winner\_vld\_q;
                                                                                                                             11
12
                                                                                                                             12
13
                                                                                                                              13
14
15
                                                                                                                                     // -- state machines --
16
17
18
       // -- one-bit state machines --
                                                                                                                              18
19
                                                                                                                                     // -- Thread Type Arb state machine --
                                                                                                                             20
20
21
       // cfs turn
                                                                                                                             21
22
                                                                                                                             22
       // - just toggle when either cfs rts is asserted
                                                                                                                                   // - arbitrates between the vertex winner and the pixel winner
23
                                                                                                                             23
                                                                                                                                     // - acknowledges the corresponding request
24
                                                                                                                              24
                                                                                                                                    // - waits until the state/status info associated with the current winner is sent to the CFS
       wire toggle_turn = arb_rts0 | arb_rts1;
25
                                                                                                                                     // before returning to pick a new thread type winner
                                              Page 15 of 28
                                                                                                                                                                            Page 16 of 28
                                                                        Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                                      Ex.\ 2097 \hbox{ - } sq\_thread\_arb.v
```

```
vtx_winner_vld_q or pix_winner_vld_q or
 2
       parameter [1:0] TTA0 = 2'b00;
                                                                                                                                       arb xfc or
       parameter [1:0] TTA1 = 2'b01;
                                                                                                                                       tta_current_state
       parameter [1:0] TTA2 = 2'b10;
                                                                                                                          4
       parameter [1:0] TTA3 = 2'b11;
                                                                                                                          5
                                                                                                                                     // default assignments
       reg [1:0] tta_current_state;
                                                                                                                                      tta_next_state = TTA0;
       reg [1:0] tta_next_state;
                                                                                                                                      type_winner = LO;
                                                                                                                                    vtx_winner_ack = LO;
10
       // state and output registers
                                                                                                                                    pix_winner_ack = LO;
11
       always @(posedge clk)
12
         begin
                                                                                                                         12
                                                                                                                                      case (tta_current_state)
13
                                                                                                                         13
            if (reset)
14
             begin
15
                                                                                                                         15
              tta current state <= TTA0;
                                                                                                                                        // - ack is connected to TB State Mem read enable, so have to
16
           end
                                                                                                                         16
                                                                                                                                         // wait until the correct phase to ack
17
                                                                                                                         17
            else
18
                                                                                                                                         if \, ( \, state\_read\_phase == arb\_type\_strap \, ) \\
             begin
                                                                                                                         18
19
               tta_current_state <= tta_next_state;
                                                                                                                         19
20
                                                                                                                         20
                                                                                                                                          if ( vtx_winner_vld_q )
                                                                                                                                                                       // simply give verts the priority
21
         end
                                                                                                                         21
                                                                                                                                          begin
22
                                                                                                                         22
                                                                                                                                           vtx_winner_ack = HI;
23
       // next state logic
                                                                                                                         23
                                                                                                                                              tta_next_state = TTA1;
24
                                                                                                                         24
25
              state_read_phase or arb_type_strap or
                                                                                                                         25
                                            Page 17 of 28
                                                                                                                                                                      Page 18 of 28
                                                                     Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                               Ex. 2097 - sq_thread_arb.v
                else if ( pix\_winner\_vld\_q )
                                                                                                                          2
 2
                                                                                                                                        if (arb xfc)
                 begin
                 pix_winner_ack = HI;
                                                                                                                                          tta_next_state = TTA0;
                     tta_next_state = TTA2;
                  end
                                                                                                                                          tta_next_state = TTA2;
                                                                                                                                       end
              end
                                                                                                                                      TTA3:
            TTA1
10
                                                                                                                         10
                                                                                                                                        // -
11
               // - wait here until the xfer from the arb to the cfs is complete
                                                                                                                         11
                                                                                                                                      tta_next_state = TTA0;
12
               // - type_winner becomes the vector_type output to the cfs, and also
                                                                                                                         12
13
               \ensuremath{/\!/} is the final mux select between vertex state and pixel state heading for the CFS
                                                                                                                         13
14
                                                                                                                         14
                                                                                                                                      endcase // case(tta_current_state)
15
               type_winner = HI;
                                                                                                                         15
16
                                                                                                                                  // - end thread type arb state machine
17
               if ( arb_xfc )
                                                                                                                         17
                                                                                                                         18
18
                tta_next_state = TTA0;
19
                                                                                                                                 wire [0:0] arb_thread_type = type_winner; // type is sent to CFS
20
               tta_next_state = TTA1;
                                                                                                                         20
21
                                                                                                                         21
             end
22
                                                                                                                         22
23
            TTA2:
                                                                                                                         23
                                                                                                                                // -- Thread Read state machine --
24
                                                                                                                         24
           begin
25
               type_winner = LO;
                                                                                                                         25
                                            Page 19 of 28
                                                                                                                                                                      Page 20 of 28
                                                                     Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                               Ex.\ 2097 \hbox{ - } sq\_thread\_arb.v
```

```
// - loads the winner register with new winner from the priority encoder
                                                                                                                                              tr_current_state <= tr_next_state;
      //- will initiate a TB state mem read, loading the state into a TB register that is sent back here to
                                                                                                                               2
                                                                                                                                          end
      /\!/ the arbiter for final muxing between vertex and pixel (e.g. tex_winner goes to both VTB and PTB,
                                                                                                                               3
                                                                                                                                       end
                                                                                                                               4
      // but the state read is specific to the TB - there's both a vtx\_tex\_ld\_state and pix\_tex\_ld\_state
                                                                                                                               5
                                                                                                                                     // next state logic
      // - TB State Mem read is enabled when there is any active request
                                                                                                                                             vtx_winner_vld_q or pix_winner_vld_q or type_winner or
                                                                                                                                             state_read_phase or arb_type_strap or
       parameter [1:0] TR0 = 2'b00;
10
                                                                                                                                            //cfs_rtr or
       parameter [1:0] TR1 = 2'b01;
11
                                                                                                                              10
                                                                                                                                             arb_xfc or
12
       parameter [1:0] TR2 = 2'b10;
                                                                                                                                             tr_current_state
13
       parameter [1:0] TR3 = 2'b11;
                                                                                                                              12
14
                                                                                                                              13
                                                                                                                                       begin
15
       reg [1:0] tr_current_state;
                                                                                                                              14
                                                                                                                                           // default assignments
16
       reg [1:0] tr_next_state;
                                                                                                                                           tr_next_state = TR0;
                                                                                                                              15
17
                                                                                                                                           ld winner = LO:
                                                                                                                              16
18
       // state and output registers
                                                                                                                              17
                                                                                                                                           arb_rts = LO;
19
       always @(posedge clk)
                                                                                                                              18
20
                                                                                                                              19
                                                                                                                                           case (tr current state)
21
                                                                                                                              20
                                                                                                                                           TRO:
22
                                                                                                                              21
                                                                                                                                          begin
23
               tr_current_state <= TR0;
                                                                                                                              22
                                                                                                                                              ld_winner = HI;
24
                                                                                                                              23
25
             else
                                                                                                                              24
                                                                                                                                              if (\ vtx\_winner\_vld\_q \ | \ pix\_winner\_vld\_q \ )
              begin
                                                                                                                              25
                                                                                                                                               begin
                                              Page 21 of 28
                                                                                                                                                                            Page 22 of 28
                                                                        Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                                      Ex. 2097 - sq_thread_arb.v
                     ld_winner = LO;
                                                                                                                               1
 2
                     tr next state = TR1;
                                                                                                                               2
                                                                                                                                          begin
                                                                                                                                              // - now the TB state read should be done (reg'd out of TB), and
                 end
              end
                                                                                                                                              \ensuremath{/\!/} this module is muxing btwn vtx and pix, with the result going to the CFS
                                                                                                                                              // - assert the rts to the cfs until the xfer is complete
             TR1
                                                                                                                                              arb rts = HI;
                // - may need to wait here a cycle depending on phase
                // - the read data load signal is generated inside the TB - it's just a flopped ack
                                                                                                                                              //if ( cfs rtr )
10
                                                                                                                              10
                                                                                                                                              if ( arb_xfc )
11
               if \ ( \ state\_read\_phase == {\sim} arb\_type\_strap \ ) \\
                                                                                                                              11
12
                                                                                                                              12
                                                                                                                                                   ld_winner = HI;
13
                                                                                                                                                   tr_next_state = TR0;
                   if ( type_winner ) ld_vtb_state = HI;
15
                           ld_ptb_state = HI;
                                                                                                                                               begin
17
                    tr_next_state = TR2;
                                                                                                                                                   tr_next_state = TR2;
18
                 end
                                                                                                                                               end
19
20
                                                                                                                              20
                begin
                                                                                                                                            end
21
                  tr_next_state = TR1;
                                                                                                                              21
22
                                                                                                                              22
                end
23
                                                                                                                              23
24
                                                                                                                              24
              end
                                                                                                                                              // - this state is not used
25
                                                                                                                              25
                                                                                                                                            tr_next_state = TR0;
                                              Page 23 of 28
                                                                                                                                                                            Page 24 of 28
                                                                       Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                                      Ex.\ 2097 \hbox{ - } sq\_thread\_arb.v
```

```
end
                                                                                                                                     // - just toggle every clock - arbitration is done every other cycle
 2
                                                                                                                                     // - this allows one cycle to pick and register the winner (from the vector of flopped requests),
 3
             endcase
                                                                                                                                      /\!/ and one cycle for the winner_q to go back and clear the request_q
 4
                                                                                                                                      aarbays @(posedge clk)
        // - end thread read state machine
                                                                                                                                           if (reset)
                                                                                                                                            ld_winner <= LO;
       // wire [0:0] tr_rts = (vtx_state_vld | pix_state_vld);
                                                                                                                                            ld_winner <= ~ld_winner;
10
                                                                                                                              10
11
                                                                                                                              11
                                                                                                                                      assign ld_winner = tr_current_state[1];
12
                                                                                                                              12
13
       always @(posedge clk)
                                                                                                                              13
14
                                                                                                                                     // - need to save the winner type until cfs acks the arb's rts
        begin
15
             if ( reset ) tr_rts <= LO;
                                                                                                                              15
16
                                                                                                                              16
                                                                                                                                      always @(posedge clk)
17
                                                                                                                              17
              case (tr rts)
                                                                                                                                      begin
18
             LO:
                                                                                                                              18
                                                                                                                                           if \ (\ reset\ ) \ type\_winner <= LO; \\
19
               tr_rts <= (vtx_state_vld | pix_state_vld);
                                                                                                                              19
20
                                                                                                                              20
                                                                                                                                            case ( type_winner )
21
               tr_rts <= ~cfs_rtr;
                                                                                                                              21
22
              endcase
                                                                                                                              22
                                                                                                                                              type_winner <= winner_thread_type;
23
         end
                                                                                                                              23
24
                                                                                                                              24
                                                                                                                                             type_winner <= ~cfs_rtr;
       // ld_winner
                                                                                                                              25
                                                                                                                                            endcase
                                              Page 25 of 28
                                                                                                                                                                            Page 26 of 28
                                                                        Ex. 2097 - sq_thread_arb.v
                                                                                                                                                                                                      Ex. 2097 - sq_thread_arb.v
         end
 2
       // - this is the ack to the thread buffer: it says the winner is valid for its current request
       // - ack the vtx TB when there's a vtx winner and the overall winner is vtx
       // - ack the pix TB when there's a pix winner and the overall winner is pix
       // - can't ack the request when we're stalled by the CFS
       /\!/ - the request, and thus the winner, will remain unchanged until the thread buff gets an ack
       /\!/ assign\ vtx\_winner\_ack =\ winner\_thread\_type\ \&\ vtx\_winner\_vld\_q\ \&\ \sim tr\_stall;
10
       // assign\ pix\_winner\_ack = \sim winner\_thread\_type\ \&\ pix\_winner\_vld\_q\ \&\ \sim tr\_stall;
                                                                                                                              10
11
12
       // - determine the overall winner between vertex winner and pixel winner
13
       // - for now verts always have priority
14
15
       wire [0:0] winner_thread_type = vtx_winner_vld_q;
16
17
18
19
20
21
22
     endmodule
23
24
25
                                              Page 27 of 28
                                                                                                                                                                            Page 28 of 28
                                                                                                                                                                                                      Ex. 2097 - sq_thread_arb.v
                                                                        Ex. 2097 - sq_thread_arb.v
```

1 'include "header.v"	1 /////////////////////////////////////
2 //	2 `include "sq_defs.v"
3 //	3
4 // \$Id: //depot/r400/devel/parts_lib/src/gfx/sq/ia/sq_input_arb.v#6 \$	4 module sq_input_arb
5 //	5 (
6 // \$Change: 42997 \$	6 vtx_req, // request from VISM
7 //	7 vtx_busy, // busy from VISM - tells arb to keep gpr write mux set to verts
8 // Copyright: Trade secret of ATI Technologies, Inc.	8 pix_req, // request from PISM
9 // © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)	9 pix_busy, // busy from PISM - tells arb to keep gpr write mux set to pixels
10 //	10
11 // All rights reserved. This notice is intended as a precaution against	11 gpr_phase, //
12 // inadvertent publication and does not imply publication or any waiver	12
13 // of confidentiality. The year included in the foregoing notice is the	13 vtx_gnt, // grant back to VISM
14 // year of creation of the work.	14 pix_gnt, // grant back to PISM
15 //	15
16 //	16 vtx_sel, // this goes to ais_output to select GPRs controls from VISM/PISM
17 ////////////////////////////////////	17
18 // sq_input_arb.v	18 clk,
19 //	19 reset
20 // - arbitrate between vertex and pixel input to the GPRs	20);
21 // - verts have priority over pixels	21
22 //	22 // parameters
23 // issues:	23
24 //-	24 parameter LO = 1'b0;
25 //	25 parameter HI = 1'b1;
Pers 1 of 10	Port 2 - £10
Page 1 of 10 Ex. 2098 - sq_input_arb.v	Page 2 of 10 Ex. 2098 - sq_input_arb.v
L parameter Y = 1 by:	
1 parameter X = 1'bx;	1
2	2 //
2 3	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4 //	2 //
2 3 4	2 //

ATI 2098 LG v. ATI IPR2015-00326

```
current\_state \mathrel{<=} IDLE;
 2
                                                                                                                                        vtx\_gnt \mathrel{<=} LO;
       // -- state machines --
                                                                                                                                        vtx_sel <= LO;
                                                                                                                                        pix_gnt <= LO;
       // input arbiter state machine
                                                                                                                                       current_state <= next_state;
       parameter IDLE = 2'b00;
                                                                                                                                       vtx_gnt <= next_vtx_gnt;
       parameter V_XFER = 2'b01;
                                                                                                                                       vtx_sel <= next_vtx_sel;
11
       parameter P_XFER = 2'b10;
                                                                                                                                       pix_gnt <= next_pix_gnt;
       parameter UNUSED = 2'b11;
                                                                                                                         12
12
13
                                                                                                                         13
                                                                                                                                  end
14
                                                                                                                         14
       reg [1:0] current_state;
15
       reg [1:0] next_state;
                                                                                                                         15
16
                                                                                                                         16
                                                                                                                                // next state logic
17
                                                                                                                        17
       reg next vtx gnt;
                                                                                                                                always @(
18
       reg next_vtx_sel;
                                                                                                                         18
                                                                                                                                     vtx_req or pix_req or vtx_busy or pix_busy or gpr_phase or
19
       reg next_pix_gnt;
                                                                                                                         19
                                                                                                                                      current_state
20
                                                                                                                        20
21
                                                                                                                        21
       // state and output registers
                                                                                                                                 begin
22
       always @(posedge clk)
                                                                                                                        22
                                                                                                                                     // default assignments
23
                                                                                                                        23
                                                                                                                                     next state = IDLE;
24
                                                                                                                        24
                                                                                                                                     next_vtx_gnt = LO;
25
                                                                                                                         25
                                                                                                                                     next_pix_gnt = LO;
                                             Page 5 of 10
                                                                                                                                                                     Page 6 of 10
                                                                      Ex. 2098 - sq_input_arb.v
                                                                                                                                                                                               Ex. 2098 - sq_input_arb.v
            next_vtx_sel = LO;
                                                                                                                                        if (vtx_busy)
 2
                                                                                                                          2
                                                                                                                                         begin
                                                                                                                                             next_vtx_sel = HI;
            case (current state)
                                                                                                                                             next_state = V_XFER;
             IDLE:
                                                                                                                                         end
                                                                                                                          6
                                                                                                                                        else
               // - assert grants based on gpr phase
               \ensuremath{/'} - gnt is reg'd out, so need to look for phase before the one that lines up
                                                                                                                                         begin
                                                                                                                                             next\_state = IDLE;
             // - the phase for pix gnt is calculated based on interp latency
               if ( vtx_req & (gpr_phase == `SQ_ID_PHASE) )
                                                                                                                                         end
10
                                                                                                                         10
                                                                                                                                      end
11
                   next_vtx_gnt = HI;
                                                                                                                         11
12
                    next_vtx_sel = HI;
                                                                                                                         12
                                                                                                                                      P_XFER:
13
                next_state = V_XFER;
                                                                                                                         13
14
                                                                                                                                        // - first check if there's another pix req (and no vtx req)
15
               else if ( pix_req & (gpr_phase == `SQ_PV_PHASE) )
                                                                                                                         15
                                                                                                                                        // - if so, grant it and stay here
16
               begin
                                                                                                                                        // - otherwise continue to hold vtx_sel low while PISM is busy
17
                   next_pix_gnt = HI;
                                                                                                                                        if (\;pix\_req \,\&\, {\sim}vtx\_req \,\&\, (gpr\_phase == `SQ\_PV\_PHASE) \,)
18
               next_state = P_XFER;
19
                                                                                                                                         begin
20
                                                                                                                         20
                                                                                                                                             next_pix_gnt = HI;
21
                                                                                                                        21
                                                                                                                                        next state = P XFER;
             end
22
                                                                                                                         22
                                                                                                                                       end
23
              V XFER:
                                                                                                                         23
                                                                                                                                        else if ( pix_busy )
24
                                                                                                                         24
                                                                                                                                         begin
25
               // - hold vtx_sel high while VISM is busy
                                                                                                                         25
                                                                                                                                             next state = P XFER;
                                             Page 7 of 10
                                                                                                                                                                     Page 8 of 10
                                                                      Ex. 2098 - sq_input_arb.v
                                                                                                                                                                                               Ex. 2098 - sq_input_arb.v
```

```
1
              end
            else
             begin
               next_state = IDLE;
            end
          endcase // case(current_state)
10
       end // always @ (*)
11
12
13
14
15
16
17
19
20
22
23
24
25
                                       Page 9 of 10
                                                                                                                                                 Page 10 of 10
                                                             Ex. 2098 - sq_input_arb.v
                                                                                                                                                                       Ex. 2098 - sq_input_arb.v
```

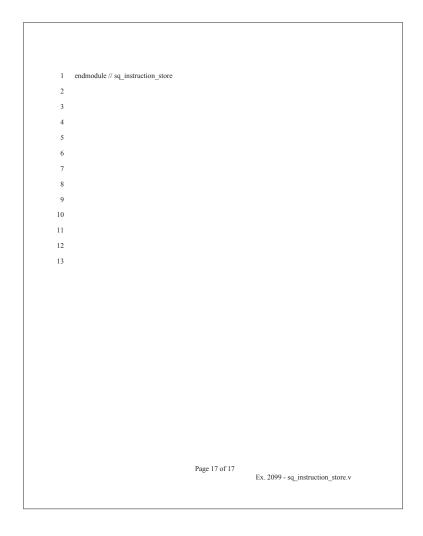
```
1 'include "header.v"
                                                                                                                    1 o_rtr, o_is_data,
2 //----
                                                                                                                    2 // Inputs
                                                                                                                    3 i_is_phase, i_is_sub_phase, i_rbi_data, i_rbi_addr, i_rts,
 4 // $Id: //depot/r400/devel/parts_lib/src/gfx/sq/is/sq_instruction_store.v#11 $
                                                                                                                    4 i_tex_addr, i_alu0_addr, i_alu1_addr, i_tex_cf_addr, i_alu0_cf_addr,
 5 //
                                                                                                                    5 i_alu1_cf_addr, i_clk, i_reset
 6 // $Change: 41218 $
 7 //
 8 //
                                                                                                                       // memory access phase control
                DFF library including plain dff, dff w/reset, dff w/clk enable,
                                                                                                                        input [1:0] i_is_phase;
10 //
                  and dff w/reset and clk enable. Reset causes rval to be loaded
                                                                                                                   10 input [1:0] i_is_sub_phase;
11 //
                  into the flop
12 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                                   12 // RBI
                 @ Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
                                                                                                                   13 input [31:0] i_rbi_data;
14 //
                                                                                                                   14 input [14:0] i_rbi_addr;
15 //
                 All rights reserved. This notice is intended as a precaution against
                                                                                                                   15 input i_rts;
16 //
                 inadvertent publication and does not imply publication or any waiver
                                                                                                                   16
                                                                                                                       output o rtr;
                                                                                                                   17
17 //
                 of confidentiality. The year included in the foregoing notice is the
18 //
                  year of creation of the work.
                                                                                                                   18
                                                                                                                        input [11:0] i_tex_addr;
20 //---
                                                                                                                   20 input [11:0] i alu0 addr;
21
                                                                                                                   21 input [11:0] i alu1 addr;
                                                                                                                   23 input [11:0] i_tex_cf_addr;
23 module sq_instruction_store
24 (/*AUTOARG*/
                                                                                                                   24 input [11:0] i_alu0_cf_addr;
25 // Outputs
                                                                                                                   25 input [11:0] i_alu1_cf_addr;
                                          Page 1 of 17
                                                                                                                                                              Page 2 of 17
                                                            Ex. 2099 - sq_instruction_store.v
                                                                                                                                                                                Ex. 2099 - sq_instruction_store.v
                                                                                                                    1 // Access to the is (instruction store) is divided into 4 phases:
 2 output [95:0] o_is_data;
                                                                                                                    2 // 0: texture instruction read
                                                                                                                    3 // 1: alu instruction read
 4
                                                                                                                    4 // The alu phase alternates between phases for alu0 and alu1.
 5 // general
                                                                                                                    5 // 2: CP write (or read for debug)
 6 input i_clk;
                                                                                                                    7 // The control flow phase is shared for accesses by alu0, alu1 and tex
                                                                                                                    8 // controlled by is_sub_phase.
 9 reg [11:0] addr, d addr;
                                                                                                                   10
11 reg [14:0] q_rbi_addr_in, d_rbi_addr_in;
12
                                                                                                                   12 always @(/*AUTOSENSE*/addr or i_alu0_addr or i_alu0_cf_addr
13 reg [95:0] wrt_data, d_wrt_data,
                                                                                                                              or i_alu1_addr or i_alu1_cf_addr or i_is_phase
14
             read data;
                                                                                                                   14
                                                                                                                              or i_is_sub_phase or i_rbi_addr or i_rbi_data or i_rts
15
                                                                                                                   15
                                                                                                                               or i tex addr or i tex cf addr or o rtr or q rbi addr in
16 reg
             we, d_we,
                                                                                                                   16
                                                                                                                               or wrt_data)
17
                                                                                                                   17 begin
              o_rtr, d_rtr;
18
                                                                                                                   18
                                                                                                                         // default values
                                                                                                                          d addr = addr;
19 wire [95:0] o_is_data = read_data;
                                                                                                                   19
                                                                                                                         d_rtr = 1'b0;
                                                                                                                   20
21 parameter TEX_PHASE = 2'd1,
                                                                                                                         d_we = 1'b0;
                                                                                                                   21
22
           ALU_PHASE = 2'd2,
                                                                                                                   22     d_wrt_data = wrt_data;
23
             CP PHASE = 2'd3.
                                                                                                                   23
                                                                                                                         case (i_is_phase)
24
             CF PHASE = 2'd0;
                                                                                                                   24
25
                                                                                                                           TEX_PHASE:
                                          Page 3 of 17
                                                                                                                                                              Page 4 of 17
                                                            Ex. 2099 - sq_instruction_store.v
                                                                                                                                                                                Ex. 2099 - sq_instruction_store.v
```

ATI 2099 LG v. ATI IPR2015-00326

```
begin
                                                                                                                                                                                                                                                                                                                         begin
  2
                              d addr
                                                           = i tex addr;
                                                                                                                                                                                                                                                                                                                           case\ (i\_is\_sub\_phase)
                               d rtr
                                                          = o_rtr;
                                                                                                                                                                                                                                                                                                                            2'b00:
                               d_wrt_data[63:32] = i_rbi_data;
                                                                                                                                                                                                                                                                                                                               d_addr = i_alu0_cf_addr;
                               d\_rbi\_addr\_in = q\_rbi\_addr\_in/3;
                                                                                                                                                                                                                                                                                                                            2'b10:
                                                                                                                                                                                                                                                                                                                               d_addr = i_alu1_cf_addr;
                     ALU_PHASE:
                                                                                                                                                                                                                                                                                                                                d_addr = i_tex_cf_addr;
                             begin
                                                                                                                                                                                                                                                                                                                          endcase // case(i_is_sub_phase)
                                                             = i_is_sub_phase[0] ? i_alu1_addr : i_alu0_addr;
11
                                                                                                                                                                                                                                                                                                                                              = o_rtr;
                               d_rtr
                                                                                                                                                                                                                                                                                                                         d_rtr
12
                               d_we
                                                           = o_rtr;
                                                                                                                                                                                                                                                                                                                          d_wrt_data[31:0] = i_rbi_data;
13
                               d_wrt_data[95:64] = i_rbi_data;
                                                                                                                                                                                                                                                                                                                 d rbi addr in = q rbi addr in/3;
14
                              d_rbi_addr_in = q_rbi_addr_in;
15
                                                                                                                                                                                                                                                                                           15
                                                                                                                                                                                                                                                                                                                          // next line needs to replace h5000 with parameter value from register addr.v
16
                                                                                                                                                                                                                                                                                           16
                                                                                                                                                                                                                                                                                                                         d_rbi_addr_in = q_rbi_addr_in - 15'h5000;
17
                     CP PHASE
                                                                                                                                                                                                                                                                                           17
                                                                                                                                                                                                                                                                                                                       end
18
                             begin
                                                                                                                                                                                                                                                                                           18
                                                                                                                                                                                                                                                                                                              endcase // case(i_is_phase)
19
                               d addr
                                                           = q rbi addr in;
                                                                                                                                                                                                                                                                                           19
20
                              d rtr
                                                    = i_rts;
                                                                                                                                                                                                                                                                                           20
                                                                                                                                                                                                                                                                                                           end // always @ (...
21
                             d wrt data[31:0] = i rbi data;
                                                                                                                                                                                                                                                                                          21
22
                              d_rbi_addr_in = i_rbi_addr;
                                                                                                                                                                                                                                                                                           22
23
                                                                                                                                                                                                                                                                                           23
                                                                                                                                                                                                                                                                                                        wire [95:0] mem_read_data;
24
                                                                                                                                                                                                                                                                                           24
                     CF_PHASE:
                                                                                                                                                                                                                                                                                                         `ifdef USE_BEHAVE_MEM
                                                                                                         Page 5 of 17
                                                                                                                                                                                                                                                                                                                                                                                                    Page 6 of 17
                                                                                                                                                    Ex. 2099 - sq_instruction_store.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                Ex. 2099 - sq_instruction_store.v
   1 dum mem p2
 2
               #(12, 96, 4096)
             u0\_dum\_mem\_p1\_4096x96
                                                                                                                                                                                                                                                                                             3 hdsd1 2048x96cm8sw0 u0 is ram
 4
                                                                                                                                                                                                                                                                                                        (/*VRGIO hdsd1 2048x96cm8sw0 wrt data mem0 rd data d addr null we 1'b1 null*/
                // Outputs
                                                                                                                                                                                                                                                                                             5 // READ/WRITE INTERFACE
                .oQ
                                              (mem read data),
                                                                                                                                                                                                                                                                                                                .CLK(i clk), // Read & Write Clock
                // Inputs
                                                                                                                                                                                                                                                                                                                 .WE(we), // Write enable
                 .iRCLK
                                            (i clk),
                                                                                                                                                                                                                                                                                                                 .OE(1'b1), // Output enable
                  iWCLK
                                                                                                                                                                                                                                                                                             9 // .ME(vdd), // Read enable
                                                                                                                                                                                                                                                                                                                .ME(~d_addr[11]), // Read enable
10
                  .iMER
                                             (1'b1).
                                                                                                                                                                                                                                                                                           10
11
                  .iMEW
                                            (1'b1),
                                                                                                                                                                                                                                                                                                                 . ADR0(d\_addr[0]), . ADR1(d\_addr[1]), . ADR2(d\_addr[2]), . ADR3(d\_addr[3]), \ // \ Address (a) = (a) - (a)
                 .iWEN
12
                                                                                                                                                                                                                                                                                                                 . ADR4(d\_addr[4]), . ADR5(d\_addr[5]), . ADR6(d\_addr[6]), . ADR7(d\_addr[7]), \ // \ Address (ADR4(d\_addr[4]), . ADR5(d\_addr[5]), . ADR5(d\_addr[5]
                                                                                                                                                                                                                                                                                                                 . ADR8(d\_addr[8]), \ . ADR9(d\_addr[9]), \ . ADR10(d\_addr[10]), \ // \ Address
                                         (d_addr),
                                                                                                                                                                                                                                                                                                                 . D0(wrt\_data[0]), . D1(wrt\_data[1]), . D2(wrt\_data[2]), . D3(wrt\_data[3]), \ // \ Write \ Data
15
                                                                                                                                                                                                                                                                                                                 . D4(wrt\_data[4]), . D5(wrt\_data[5]), . D6(wrt\_data[6]), . D7(wrt\_data[7]), \ // \ Write \ Data
                                             (wrt_data));
                                                                                                                                                                                                                                                                                                                 . D8(wrt\_data[8]), . D9(wrt\_data[9]), . D10(wrt\_data[10]), . D11(wrt\_data[11]), \ //\ Write\ Data
17
             `else // !ifdef USE_BEHAVE_MEM
                                                                                                                                                                                                                                                                                                       .D12(wrt_data[12]), .D13(wrt_data[13]), .D14(wrt_data[14]), .D15(wrt_data[15]), // Write Data
18
            // due to the speed of this ram, it had to be split into two 2048x96 rams
                                                                                                                                                                                                                                                                                                        .D16(wrt_data[16]), .D17(wrt_data[17]), .D18(wrt_data[18]), .D19(wrt_data[19]), \ /\!/ Write Data
19
20
         wire vdd = 1'b1;
                                                                                                                                                                                                                                                                                                                 . D20(wrt\_data[20]), \ . D21(wrt\_data[21]), \ . D22(wrt\_data[22]), \ . D23(wrt\_data[23]), \ // \ Write
21
         wire vss = 1'b0;
                                                                                                                                                                                                                                                                                                       . D24(wrt\_data[24]), .D25(wrt\_data[25]), .D26(wrt\_data[26]), .D27(wrt\_data[27]), \ // \ Write Data
22
          wire [95:0] mem0 rd data;
23
          wire [95:0] mem1 rd data;
                                                                                                                                                                                                                                                                                                                . D28(wrt\_data[28]), \ . D29(wrt\_data[29]), \ . D30(wrt\_data[30]), \ . D31(wrt\_data[31]), \ // \ Write
24
                                                                                                                                                                                                                                                                                                                . D32(wrt\_data[32]), \ . D33(wrt\_data[33]), \ . D34(wrt\_data[34]), \ . D35(wrt\_data[35]), \ // \ Write
             Page 7 of 17
                                                                                                                                                                                                                                                                                                                                                                                                    Page 8 of 17
                                                                                                                                                    Ex. 2099 - sq instruction store.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                Ex. 2099 - sq_instruction_store.v
```

1 .D36(wrt_data[36]), .D37(wrt_data[37]), .D38(wrt_data[38]), .D39(wrt_data[39]), // Write 2 Data	1 .Q4(mem0_rd_data[4]), .Q5(mem0_rd_data[5]), .Q6(mem0_rd_data[6]), 2 .Q7(mem0_rd_data[7]), // Read Data
3D40(wrt_data[40]), .D41(wrt_data[41]), .D42(wrt_data[42]), .D43(wrt_data[43]), // Write 4Data	3 .Q8(mem0_rd_data[8]), .Q9(mem0_rd_data[9]), .Q10(mem0_rd_data[10]), 4 .Q11(mem0_rd_data[11]), // Read Data
5 .D44(wrt_data[44]), .D45(wrt_data[45]), .D46(wrt_data[46]), .D47(wrt_data[47]), // Write 6 .Data	5 .Q12(mem0_rd_data[12]), .Q13(mem0_rd_data[13]), .Q14(mem0_rd_data[14]), 6 .Q15(mem0_rd_data[15]), // Read Data
7	7 .Q16(mem0_rd_data[16]), .Q17(mem0_rd_data[17]), .Q18(mem0_rd_data[18]), 8 .Q19(mem0_rd_data[19]), // Read Data
9	9 .Q20(mem0_rd_data[20]), .Q21(mem0_rd_data[21]), .Q22(mem0_rd_data[22]), 10 .Q23(mem0 rd_data[23]), // Read Data
11	11 .Q24(mem0_rd_data[24]), .Q25(mem0_rd_data[25]), .Q26(mem0_rd_data[26]), 12 .Q27(mem0 rd_data[27]), // Read Data
13	13 .Q28(mem0_rd_data[28]), .Q29(mem0_rd_data[29]), .Q30(mem0_rd_data[30]), 14 .Q31(mem0_rd_data[31]), // Read Data
15D64(wrt_data[64]), .D65(wrt_data[65]), .D66(wrt_data[66]), .D67(wrt_data[67]), // Write 16Data	15 .Q32(mem0_rd_data[32]), .Q33(mem0_rd_data[33]), .Q34(mem0_rd_data[34]), .Q35(mem0_rd_data[35]), // Read Data
17D68(wrt_data[68]), .D69(wrt_data[69]), .D70(wrt_data[70]), .D71(wrt_data[71]), // Write 18Data	17 .Q36(mem0_rd_data[36]), .Q37(mem0_rd_data[37]), .Q38(mem0_rd_data[38]), 18 .Q39(mem0_rd_data[39]), // Read Data
19D72(wrt_data[72]), .D73(wrt_data[73]), .D74(wrt_data[74]), .D75(wrt_data[75]), // Write 20Data	19 .Q40(mem0_rd_data[40]), .Q41(mem0_rd_data[41]), .Q42(mem0_rd_data[42]), 20 .Q43(mem0_rd_data[43]), // Read Data
21D76(wrt_data[76]), .D77(wrt_data[77]), .D78(wrt_data[78]), .D79(wrt_data[79]), // Write 22Data	21 .Q44(mem0_rd_data[44]), .Q45(mem0_rd_data[45]), .Q46(mem0_rd_data[46]), 22 .Q47(mem0_rd_data[47]), // Read Data
23	23 .Q48(mem0_rd_data[48]), .Q49(mem0_rd_data[49]), .Q50(mem0_rd_data[50]), 24 .Q51(mem0_rd_data[51]), // Read Data
25D84(wrt_data[84]), .D85(wrt_data[85]), .D86(wrt_data[86]), .D87(wrt_data[87]), // Write 26Data	25 .Q52(mem0_rd_data[52]), // Read Data 26 .Q52(mem0_rd_data[52]), .Q53(mem0_rd_data[53]), .Q54(mem0_rd_data[54]), 27 .Q52(mem0_rd_data[55]), // Read Data
27D88(wrt_data[88]), .D89(wrt_data[89]), .D90(wrt_data[90]), .D91(wrt_data[91]), // Write 28Data	27 .Q56(mem0_rd_data[56]), //Read Data 28 .Q59(mem0_rd_data[58]), //Read Data
29D92(wrt_data[92]), .D93(wrt_data[93]), .D94(wrt_data[94]), .D95(wrt_data[95]), // Write 30Data	29 .Q60(mem0_rd_data[60]), //Read Data 30 .Q63(mem0 rd_data[63]), //Read Data
31 .Q0(mem0_rd_data[0]), .Q1(mem0_rd_data[1]), .Q2(mem0_rd_data[2]), 32 .Q3(mem0_rd_data[3]), // Read Data	31 .Q64(mem0_rd_data[64]), .Q65(mem0_rd_data[65]), .Q66(mem0_rd_data[66]), 32 .Q67(mem0 rd_data[67]), // Read Data
Page 9 of 17 Ex. 2099 - sq_instruction_store.v	Page 10 of 17 Ex. 2099 - sq_instruction_store.v
1	(/*VRGIO hdsd1_2048x96cm8sw0 wrt_data mem1_rd_data d_addr null we 1'b1 null*/ /*READ/WRITE INTERFACE
29 hdsd1_2048x96cm8sw0 u1_is_ram Page 11 of 17	28 .D40(wrt_data[40]), .D41(wrt_data[41]), .D42(wrt_data[42]), .D43(wrt_data[43]), // Write 29 Data
Page 11 of 17	Page 12 of 17

```
.Q12(mem1_rd_data[12]), .Q1
.Q15(mem1_rd_data[15]), // Read Data
                           . D44(wrt\_data[44]), \ . D45(wrt\_data[45]), \ . D46(wrt\_data[46]), \ . D47(wrt\_data[47]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    .Q13(mem1_rd_data[13]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .Q14(mem1_rd_data[14]),
                 . D48(wrt\_data[48]), .D49(wrt\_data[49]), .D50(wrt\_data[50]), .D51(wrt\_data[51]), \ // \ Write Data
                                                                                                                                                                                                                                                                                                                                                                    .Q16(mem1\_rd\_data[16]), \qquad .Q17(mem1\_rd\_data[17]), \\ .Q19(mem1\_rd\_data[19]), \ // \ Read \ Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q18(mem1_rd_data[18]),
                                                                                                                                                                                                                                                                                                                                                                     .Q20(mem1\_rd\_data[20]), \qquad .Q21(mem1\_rd\_data[21]), \\ .Q23(mem1\_rd\_data[23]), \ //\ Read\ Data
                           . D52(wrt\_data[52]), \ . D53(wrt\_data[53]), \ . D54(wrt\_data[54]), \ . D55(wrt\_data[55]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q22(mem1_rd_data[22]),
                                                                                                                                                                                                                                                                                                                                                                     .Q24(mem1_rd_data[24]), .Q2:
.Q27(mem1_rd_data[27]), // Read Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    .Q25(mem1_rd_data[25]),
                           . D56(wrt\_data[56]), \ . D57(wrt\_data[57]), \ . D58(wrt\_data[58]), \ . D59(wrt\_data[59]), \ \# Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .Q26(mem1 rd data[26]),
                           . D60(wrt\_data[60]), \ . D61(wrt\_data[61]), \ . D62(wrt\_data[62]), \ . D63(wrt\_data[63]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                .Q28(mem1 rd data[28]).
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    .Q29(mem1_rd_data[29]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .Q30(mem1_rd_data[30]),
 10
                                                                                                                                                                                                                                                                                                                                                                     .Q31(mem1_rd_data[31]), // Read Data
                                                                                                                                                                                                                                                                                                                                                                     .Q32(mem1_rd_data[32]), .Q3
.Q35(mem1_rd_data[35]), // Read Data
                           . D64(wrt\_data[64]), \ . D65(wrt\_data[65]), \ . D66(wrt\_data[66]), \ . D67(wrt\_data[67]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                   .Q33(mem1_rd_data[33]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q34(mem1_rd_data[34]),
 12
                                                                                                                                                                                                                                                                                                                                                                     .Q36(mem1\_rd\_data[36]), \qquad .Q37(mem1\_rd\_data[37]), \\ .Q39(mem1\_rd\_data[39]), \ //\ Read\ Data
                           . D68(wrt\_data[68]), \ . D69(wrt\_data[69]), \ . D70(wrt\_data[70]), \ . D71(wrt\_data[71]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q38(mem1 rd data[38]),
15
16
                           .D72(wrt_data[72]), .D73(wrt_data[73]), .D74(wrt_data[74]), .D75(wrt_data[75]), // Write
                                                                                                                                                                                                                                                                                                                                                                     .Q40(mem1_rd_data[40]), .Q4
.Q43(mem1_rd_data[43]), // Read Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                     .Q41(mem1 rd data[41]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .Q42(mem1 rd data[42]),
                                                                                                                                                                                                                                                                                                                                                                     .Q44(meml_rd_data[44]), .Q4:
.Q47(meml_rd_data[47]), // Read Data
                           . D76(wrt\_data[76]), \ . D77(wrt\_data[77]), \ . D78(wrt\_data[78]), \ . D79(wrt\_data[79]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    .Q45(mem1_rd_data[45]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .O46(mem1 rd data[46]).
 18
                           . D80(wrt\_data[80]), \ . D81(wrt\_data[81]), \ . D82(wrt\_data[82]), \ . D83(wrt\_data[83]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                     .Q48(mem1_rd_data[48]), .Q4
.Q51(mem1_rd_data[51]), // Read Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    .Q49(mem1_rd_data[49]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q50(mem1_rd_data[50]),
 20
                           .D84(wrt\_data[84]),\ .D85(wrt\_data[85]),\ .D86(wrt\_data[86]),\ .D87(wrt\_data[87]),\ //\ Write
                                                                                                                                                                                                                                                                                                                                                                     .Q52(mem1\_rd\_data[52]), \qquad .Q53(mem1\_rd\_data[53]), \\ .Q55(mem1\_rd\_data[55]), \ // \ Read \ Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q54(mem1_rd_data[54]),
23
24
                           . D88(wrt\_data[88]), \ . D89(wrt\_data[89]), \ . D90(wrt\_data[90]), \ . D91(wrt\_data[91]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                                                                                                                   .Q57(mem1_rd_data[57]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q58(mem1_rd_data[58]),
                                                                                                                                                                                                                                                                                                                                                                      .Q59(mem1 rd data[59]), // Read Data
25
26
                           . D92(wrt\_data[92]), \ . D93(wrt\_data[93]), \ . D94(wrt\_data[94]), \ . D95(wrt\_data[95]), \ // \ Write
                                                                                                                                                                                                                                                                                                                                                                     .Q60(mem1\_rd\_data[60]), \qquad .Q61(mem1\_rd\_data[61]), \\ .Q63(mem1\_rd\_data[63]), \ // Read \ Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .Q62(mem1 rd data[62]),
                 .Q0(mem1\_rd\_data[0]), \\ .Q3(mem1\_rd\_data[3]), \ //\ Read\ Data \\ \\ .Q3(mem1\_rd\_data[1]), \\ .Q3(mem1\_rd\_data[1]), \\ .Q4(mem1\_rd\_data[1]), \\ .Q5(mem1\_rd\_data[1]), \\ .Q6(mem1\_rd\_data[1]), \\ .Q7(mem1\_rd\_data[1]), \\ .Q8(mem1\_rd\_data[1]), \\ .Q9(mem1\_rd\_data[1]), \\ .Q9(mem1\_
                                                                                                                                                                                                                                                                                                                                                                     .Q64(mem1\_rd\_data[64]), \qquad .Q6\\ .Q67(mem1\_rd\_data[67]), \ // \ Read \ Data
27
28
                                                                                                                                                                                                            .Q2(mem1_rd_data[2]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                    .Q65(mem1_rd_data[65]),
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          .Q66(mem1_rd_data[66]),
                  .Q4(meml\_rd\_data[4]), \\ .Q7(meml\_rd\_data[7]), \ //\ Read\ Data
                                                                                                                                                                                                            .Q6(mem1_rd_data[6]),
                                                                                                                                                                                                                                                                                                                                                                     .Q68(meml\_rd\_data[68]), \qquad .Q69(meml\_rd\_data[69]), \\ .Q71(meml\_rd\_data[71]), \; // \, Read \; Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q70(mem1_rd_data[70]),
                 .Q8(mem1\_rd\_data[8]), \qquad .Q9(mem1\_rd\_data[9]), \\ .Q11(mem1\_rd\_data[11]), \ //\ Read\ Data
                                                                                                                                                                                                                                                                                                                                                                     .Q72(mem1\_rd\_data[72]), \qquad .Q73(mem1\_rd\_data[73]), \\ .Q75(mem1\_rd\_data[75]), \ // \ Read \ Data
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           .Q74(mem1 rd data[74]),
                                                                                                                             Page 13 of 17
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                Page 14 of 17
                                                                                                                                                                                  Ex. 2099 - sq_instruction_store.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        Ex. 2099 - sq_instruction_store.v
                                                                                                                .Q77(mem1_rd_data[77]),
                            .Q76(mem1 rd data[76]),
                                                                                                                                                                                                      .Q78(mem1 rd data[78]),
                                                                                                                                                                                                                                                                                                                                                                           if (i reset)
                .Q79(mem1_rd_data[79]), // Read Data
                                                                                                                                                                                                                                                                                                                                                       2
                                                                                                                                                                                                                                                                                                                                                                             begin
                            .O80(mem1 rd data[80]).
                                                                                                               .Q81(mem1_rd_data[81]),
                                                                                                                                                                                                      .Q82(mem1 rd data[82]),
                  .Q83(mem1_rd_data[83]), // Read Data
                                                                                                                                                                                                                                                                                                                                                                                        we
                                                                                                                                                                                                                                                                                                                                                                                                                  <= 1'b0:
                  .Q84(meml_rd_data[84]), .Q85(meml_rd_data[85]), .Q87(meml_rd_data[87]), // Read Data
                                                                                                                                                                                                      .Q86(mem1_rd_data[86]),
                                                                                                                                                                                                                                                                                                                                                                                      addr
                                                                                                                                                                                                                                                                                                                                                                                                                  <= 12'd0:
                                                                                                                                                                                                                                                                                                                                                                                        read data <= 96'd0:
                  .Q88(meml_rd_data[88]), .Q89(meml_rd_data[89]), .Q91(meml_rd_data[91]), // Read Data
                                                                                                                                                                                                      .Q90(mem1_rd_data[90]),
                                                                                                                                                                                                                                                                                                                                                                                                               <= 1'b0:
                                                                                                                                                                                                                                                                                                                                                                                        wrt data <= 96'd0;
                 .Q92(mem1_rd_data[92]), .Q9
.Q95(mem1_rd_data[95]), // Read Data
                                                                                                               .Q93(mem1 rd data[93]),
                                                                                                                                                                                                      .Q94(mem1 rd data[94]),
10
                                                                                                                                                                                                                                                                                                                                                                                        q_rbi_addr_in <= 12'd0;
11
                         // READ/WRITE TEST SIGNALS
                                                                                                                                                                                                                                                                                                                                                       9
                                                                                                                                                                                                                                                                                                                                                                              end
12
                           .BISTE(vss).
                                                                                                                                                                                                                                                                                                                                                    10
                           .TWE(vss),
13
                                                                                                                                                                                                                                                                                                                                                    11
14
                           TOE(vss)
                                                                                                                                                                                                                                                                                                                                                    12
15
                                                                                                                                                                                                                                                                                                                                                                                                                  <= d_addr;
                            . TADR0(d\_addr[0]), \quad . TADR1(d\_addr[1]), \quad . TADR2(d\_addr[2]), \quad . TADR3(d\_addr[3]), \quad // (d\_addr[0]), \quad . TADR3(d\_addr[0]), \quad . 
                                                                                                                                                                                                                                                                                                                                                                                       read_data <= mem_read_data;
                                                                                                                                                                                                                                                                                                                                                                                                             <= d_rtr;
                           . TADR4(d\_addr[4]), \;\; . TADR5(d\_addr[5]), \;\; . TADR6(d\_addr[6]), \;\; . TADR7(d\_addr[7]), \;\; // (d\_addr[6]), \;\; . TADR7(d\_addr[7]), \;\; // (d\_addr[6]), \;\; . TADR7(d\_addr[7]), \;\; // (d\_addr[8]), \;\; . TADR7(d\_addr[8]), \;\; . TADR7(d
                                                                                                                                                                                                                                                                                                                                                                                        wrt_data <= d_wrt_data;
20
                           .TADR8(d_addr[8]), .TADR9(d_addr[9]), .TADR10(d_addr[10]), // Write Test Address
                                                                                                                                                                                                                                                                                                                                                    17
                                                                                                                                                                                                                                                                                                                                                                                    q_rbi_addr_in <= d_rbi_addr_in;
21
                           .RM0(vss), .RM1(vss), .RM2(vss), .RM3(vss), // Read Margin
                                                                                                                                                                                                                                                                                                                                                    18
22
                           .AWT(vss)
                                                                                                                                                                                                                                                                                                                                                     19
                                                                                                                                                                                                                                                                                                                                                                       end
23
                                                                                                                                                                                                                                                                                                                                                    20
24
                  `endif
                                                                                                                                                                                                                                                                                                                                                    21
25
                                                                                                                                                                                                                                                                                                                                                                // Local Variables:
26
                // register instantiation
                                                                                                                                                                                                                                                                                                                                                                // verilog-library-directories:("." "../../common/")
                                                                                                                                                                                                                                                                                                                                                    23
27
                always @(posedge i_clk)
                                                                                                                                                                                                                                                                                                                                                    24
28
                    begin
                                                                                                                                                                                                                                                                                                                                                     25
                                                                                                                             Page 15 of 17
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                Page 16 of 17
                                                                                                                                                                                  Ex. 2099 - sq instruction store.y
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        Ex. 2099 - sq_instruction_store.v
```



```
// instr_ptr:13, exec_cnt:4, export_id:1
                                                                                                           2 'define SQ CFS STATE WIDTH 25 // max of the above
                        // first:1, instr_ptr:12, resource:1, valid_bits:64, gpr_base:7,
    context id:3
                                                                                                          4 // define SQ_TEX_CFS_STATE_WIDTH 18 // number of bits in the tex thread CFS state
    `define SQ_PIX_CTL_PKT_WIDTH 184 // number of bits in the pixel control packet
     /\!/ \  \  first:1, \  \  base\_ptr:12, \  \  resource:1, \  \  lod\_correct:96, \  \  valid\_bits:64, \\ gpr\_base:7, context\_id:3
                                                                                                          5 // define SQ_ALU_CFS_STATE_WIDTH 25 // number of bits in the ALU thread state
    `define SQ_CTL_PKT_WIDTH 184 // number of bits in the pixel control packet
                                                                                                          7 // Status Regs
                                                                                                           8 'define SQ_VTX_STATUS_WIDTH 22 // number of bits in the vertex thread status
10 // State Mem
                                                                                                                                  // thread id:4, status:14
11 'define SQ_VTX_STATE_WIDTH 99 // ism + cfs state
                                                                                                          10 'define SQ_PIX_STATUS_WIDTH 22 // number of bits in the pixel thread status
12 'define SO PIX STATE WIDTH 236 //
                                                                                                                // thread id:4, status:14
13 'define SQ_STATE_WIDTH 236 // max of the above
                                                                                                          12 'define SQ_STATUS_WIDTH 22 // max of the above
                                                                                                          13
15 // ISM State Mem
                                                                                                          14 // instruction sequencer type - compared to inst store phase and alu phase
     `define SQ_VTX_ISM_STATE_WIDTH 74 // number of bits in the vertex thread ISM
                                                                                                              `define SQ_TEX_STRAP \, 3'b010 \, // - this is what {is_phase, alu_phase} should match
                       // valid_bits:64, gpr_base:7, context_id:3 (note that valid_bits will
18
19
                                                                                                          17 'define SQ_ALU0_STRAP 3'b100 \mbox{//} alu_phase is also tied low on the tex instance)
    move to alu state)
20
    'define SQ_PIX_ISM_STATE_WIDTH 218 // number of bits in the pixel thread state
                                                                                                          18 'define SQ_ALU1_STRAP 3'b101
                                                                                                          19
21
                        // lod_correct:144, valid_bits:64, gpr_base:7, context_id:3
                                                                                                          20 // instr store phase
     `define SQ_ISM_STATE_WIDTH 218 // max of the above
                                                                                                         21 'define SO IS CES PHASE 2'b00
                                                                                                         22 'define SQ IS TEX PHASE 2'b01
                                                                                                         23 'define SQ_IS_ALU_PHASE 2'b10
    'define SQ_VTX_CFS_STATE_WIDTH 25 // number of bits in the vertex thread CFS
                                                                                                         24 'define SQ_IS_CP_PHASE 2'b11
                        // instr_ptr:13, exec_cnt:4, param_ptr:7, export_id:1
28 'define SQ_PIX_CFS_STATE_WIDTH 18 // number of bits in the pixel thread state
                                                                                                          26 // instr store CFS sub phase
                                       Page 1 of 4
                                                                                                                                                  Page 2 of 4
                                                                 Ex. 2100 - sq_defs.v
                                                                                                                                                                            Ex. 2100 - sq_defs.v
1 'define SQ_IS_ALU0_SUBPHASE 2'b00
                                                                                                           1 'define SQ_PB_WIDTH
                                                                                                                                         53 // width of the pointer buffer
2 'define SQ IS TEX0 SUBPHASE 2'b01
                                                                                                           2
3 'define SQ IS ALU1 SUBPHASE 2'b10
                                                                                                           3 // vector type: 1 = VTX, 0 = PIX
4 'define SQ_IS_TEX1_SUBPHASE 2'b11 // for tex, only care if subphase lsb == 1
                                                                                                           4 'define SO VTX 1'b1
                                                                                                           5 'define SQ_PIX 1'b0
6 // CFS phase
    'define SQ_CFS_ALU0_PHASE 2'b00
    'define SQ_CFS_TEX0_PHASE 2'b01
9 'define SQ_CFS_ALU1_PHASE 2'b10
10 'define SQ_CFS_TEX1_PHASE 2'b11
                                            // for tex, only care if phase lsb == 1
12 // gpr write phase
13 'define SQ_ID_PHASE 2'b00 // Input Data gpr write phase
14 'define SQ_FD_PHASE 2'b01 // Fetch Data gpr write phase
15 'define SQ_PV_PHASE 2'b10 // Vector Result (PV) gpr write phase
16 'define SQ_PS_PHASE 2'b11 // Scalar Result (PS) gpr write phase
17
18 // gpr read phase
19 'define SQ SRCB PHASE 2'b00 // source B gpr read phase
20 'define SQ_SRCC_PHASE 2'b01 // source C gpr read phase
21 'define SQ_FA_PHASE 2'b10 // fetch address gpr read phase
22 'define SQ_SRCA_PHASE 2'b11 // source A gpr read phase
25 `define SQ_SC_DATA_WIDTH 54 // width of the scan converter input data bus
                                       Page 3 of 4
                                                                                                                                                  Page 4 of 4
                                                                 Ex. 2100 - sq_defs.v
                                                                                                                                                                            Ex. 2100 - sq_defs.v
```

ATI 2100 LG v. ATI IPR2015-00326

```
1 'include "header.v"
                                                                                                                      1 // - the winning thread id is used to read the State Mem and the Status Registers so that
                                                                                                                      2\ \ //\  the selected thread info can be sent to the associated Control Flow Sequencer (CFS) (via the 3\ \  arb)
 4 \hspace{0.5cm} \textit{//} \$ Id: \textit{//} depot/r400/devel/parts\_lib/src/gfx/sq/ss/sq\_thread\_buff.v\#27 \$
                                                                                                                      5 // - the sq top level will contain two instances of the thread buffer (one for verts, on for pixels)
 5 //
                                                                                                                      6 \hspace{0.4cm} // - the tex requests from both TBs go to the tex thread arbiter, and the alu requests from both 7\hspace{0.4cm} TBs
 6 // $Change: 44010 $
 7 //
 8 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                                      9 \hspace{0.5cm} \textit{//-} so there are 4 total winners from two arbiters: } vtx\_tex, pix\_tex, vtx\_alu, and pix\_alu
                  © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
                                                                                                                     10 // - each arbiter then does the final muxing between vtx and pix
10 //
11 //
                 All rights reserved. This notice is intended as a precaution against
                                                                                                                    12 // issues
                inadvertent publication and does not imply publication or any waiver
                                                                                                                    13 // -
13 //
                of confidentiality. The year included in the foregoing notice is the
14 //
                year of creation of the work.
                                                                                                                    15 //
                                                                                                                    16
                                                                                                                    17 module sq_thread_buff
18 (
18 // sq thread buff.v
                                                                                                                    19 thread_type_strap, // a strap that tells this module if it's a vertex or pixel thread buffer
                                                                                                                    20 state read phase, // share read access between tex and alu arbs
20 // - this is the clause-less thread buffer (a.k.a. reservation station)
                                                                                                                                          // share write (update) access between the tex and alu CFSs
21 // - contains Thread State Memory and Thread Status Registers
                                                                                                                    22
22 // - each line (thread) has both a texture and an alu request output (only one can
                                                                                                                    23 // inputs from local registers
                                                                                                                                                    // connected to SQ_PROGRAM_CNTL.VS_NUM_REG (6 bits)
                                                                                                                     24 num_reg_set,
25 (or PS_NUM_REG)
24 // - tex requests go to the tex arb, and a tex winner is returned
25 // - alu requests go to the alu arb, and a alu_winner is returned
                                           Page 1 of 49
                                                                                                                                                                Page 2 of 49
                                                                  Ex. 2101 - sq thread buff.v
                                                                                                                                                                                       Ex. 2101 - sq thread buff.v
 1 // control packet input (from ISM) - initial values for state and status
                                                                                                                      1 acfs1_state,
                                                                                                                                              // state returned from cfs
                                                                                                                      2 acfs1 status,
      ism rts,
                        // control packet rts
                                                                                                                                             // status returned from cfs
      ism_lod_correct, // state (pix only)
 5 ism valid bits, // state
                                                                                                                     5 // tex thread arbiter interface
 6 ism_gpr_base, // state
                                                                                                                      6 tex_req_q, // tex request from every thread in the buffer
 7 ism_context_id, // state
                                                                                                                      7 tex_winner_q, // tex winner from arbiter
 8 ism resource, // status: resource bit : tex=1, alu=0
                                                                                                                     8 tex_winner_ack, // tex winner valid (request acknowledge) from tex arbiter
      ism first thread, // status: first thread of a new state
                                                                                                                                             // winning state read from State Mem back to tex arbiter
                                                                                                                           tex state q,
                         // rtr when not full AND not doing a CFS update
                                                                                                                                             // winning status read from Status Regs back to tex arbiter
11
                                                                                                                    11
                                                                                                                    12 TP_SQ_data_rdy, // data ready (done) indicator from TPC
12 // tex control flow seq update of state and status
13 tcfs_update, // load updated status info from CFS
                                                                                                                    13 TP_SQ_type,
                                                                                                                                                 // the vector type: pixel=0, vertex=1
14 tcfs_thread_type, // the vector type: pixel=0, vertex=1
                                                                                                                    14 TP SQ thread id, //
15
      tcfs state,
                        // state returned from cfs
                                                                                                                    15
                                                                                                                     16
                                                                                                                    17
                                                                                                                           alu_req_q, // alu req from every thread
18 // alu control flow seq update of state and status
                                                                                                                    18 alu winner q, // alu winner from arbiter
                               // load updated status info from CFS
                                                                                                                     19 alu_winner_ack, // alu winner valid from alu arbiter
20 acfs0_thread_type, // the vector type: pixel=0, vertex=1
                                                                                                                    20 alu_state_q, // winning state read from State Mem
                                                                                                                    21
21 acfs0_state, // state returned from cfs
                                                                                                                           alu_status_q,
                                                                                                                                            // winning status read from Status Regs
       acfs0_status,
                       // status returned from cfs
23
                                                                                                                     23 ais0 done,
                                                                                                                                             // done indicator from AIS0
24 acfs1 update.
                               // load updated status info from CFS
                                                                                                                    24 ais0 thread type, // the vector type: pixel=0, vertex=1
      acfs1_thread_type, // the vector type: pixel=0, vertex=1
                                                                                                                     25 ais0_thread_id, //
                                           Page 3 of 49
                                                                                                                                                               Page 4 of 49
                                                                  Ex. 2101 - sq thread buff.v
                                                                                                                                                                                       Ex. 2101 - sq thread buff.v
```

ATI 2101 LG v. ATI IPR2015-00326

```
2
                                                                                                                        2
       ais1 done.
                          // done indicator from AIS1
                                                                                                                              busy,
                                                                                                                                          // TB is busy when there are any threads in the buffer
       ais\,l\_thread\_type, \quad \textit{// the vector type: pixel=0, vertex=1}
                                                                                                                        3
                                                                                                                              clk.
       ais1_thread_id, //
       // SX export buffer availability
       u0_SX_SQ_exp_count_rdy,// avaliability info is valid
       u0_SX_SQ_exp_pos_avail,// position available from SX
                                                                                                                              // -- parameters --
      u0\_SX\_SQ\_exp\_buf\_avail,\!/\!/ buffer available from SX (0 to 127 2kbit buffers(), 2kbit = 32bits for 64 pixels)
10
                                                                                                                              parameter STATE_WIDTH = 8;
11
                                                                                                                              parameter ISM_STATE_WIDTH = 8;
12
       ul_SX_SQ_exp_count_rdy,
                                                                                                                              parameter CFS_STATE_WIDTH = 8;
13
       ul SX SQ exp pos avail,
                                                                                                                       13
                                                                                                                              //parameter ALU_STATE_WIDTH = 8;
14
       ul SX SQ exp buf avail,
15
                                                                                                                       15
                                                                                                                              parameter STATUS_WIDTH = 8;
16
       // outputs from exit SM to constant stores and gpr alloc
                                                                                                                       16
17
       // - this needs to be checked ..
                                                                                                                       17
                                                                                                                              parameter TB_DEPTH = 16; // number of locations
18
       state_change, // a pulse high indicates that the state exiting the SS has changed
                                                                                                                             parameter TB_ADDR_WIDTH = 4; // log2 (number of locations rounded to nearest power of 2)
19
                         // the state that has finished (because a new state has emerged)
                                                                                                                       20
20
       dealloc req,
                         // request to deallocate GPRs
21
       dealloc_space,
                        // number of locations to dealloc (from local gfx reg)
                                                                                                                       21
                                                                                                                              parameter TID_WIDTH = 6; // number ob bits in the thread ID
22
       dealloc_ack,
                         // the dealloc request has been acknowleged
                                                                                                                       22
23
                                                                                                                       23
                                                                                                                              parameter LO = 1'b0;
24
       pop_thread,
                         // syncs vtx shader with pix input (latter must wait for former)
                                                                                                                       24
                                                                                                                              parameter HI = 1'b1;
25
                                                                                                                       25
                                                                                                                              parameter X = 1'bx;
       param_cache_wptr_q, // input to status regs
                                            Page 5 of 49
                                                                                                                                                                   Page 6 of 49
                                                                   Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                          Ex. 2101 - sq_thread_buff.v
                                                                                                                                                       tcfs_thread_type;
                                                                                                                              input [0:0]
                                                                                                                              input [CFS_STATE_WIDTH-1:0] tcfs_state;
                                                                                                                              input [STATUS_WIDTH-1:0] tcfs_status;
       // -- ios --
       input [0:0] thread_type_strap;
                                                                                                                                                       acfs0 update;
                                                                                                                              input [0:0]
       input [0:0] state_read_phase;
                                                                                                                              input [0:0]
                                                                                                                                                       acfs0 thread type:
                                                                                                                              input [CFS_STATE_WIDTH-1:0] acfs0_state;
       input [1:0] cfs_phase;
                                                                                                                              input [STATUS_WIDTH-1:0] acfs0_status;
10
       input [8*6-1:0] num_reg_set;
                                                                                                                       10
11
                                                                                                                       11
12
                                                                                                                       12
                                                                                                                              input [0:0]
                                                                                                                                                       acfs1_update;
13
       input [00:0] ism_rts;
                                                                                                                                                       acfs1_thread_type;
                                                                                                                              input [CFS_STATE_WIDTH-1:0] acfs1_state;
       input [143:0] ism_lod_correct;
15
                                                                                                                              input [STATUS_WIDTH-1:0] acfs1_status;
       input [11:0] ism_instr_ptr;
16
       input [63:0] ism_valid_bits;
17
       input [06:0] ism_gpr_base;
                                                                                                                       17
18
       input [02:0] ism_context_id;
                                                                                                                       18
                                                                                                                              output [TB_DEPTH-1:0] tex_req_q;
19
       input [00:0] ism_resource;
                                                                                                                              output [STATE_WIDTH-1:0] tex_state_q;
                                                                                                                              output [STATUS_WIDTH-1:0] tex_status_q;
20
       input [00:0] ism first thread;
                                                                                                                       20
21
                                                                                                                       21
22
                                                                                                                       22
                                                                                                                              input [TB_ADDR_WIDTH-1:0] tex_winner_q;
       output [0:0] tb rtr;
23
                                                                                                                       23
                                                                                                                              input [0:0]
                                                                                                                                              tex winner ack:
24
                                                                                                                       24
25
       input [0:0]
                                 tcfs_update;
                                                                                                                              input [0:0]
                                                                                                                                                       TP_SQ_data_rdy;
                                                                                                                                                                   Page 8 of 49
                                            Page 7 of 49
                                                                   Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                          Ex. 2101 - sq_thread_buff.v
```

```
input [0:0]
                        TP SQ type;
                                                                                                                         //tri1 [0:0] u1_SX_SQ_exp_pos_avail;
       input [TID_WIDTH-1:0] TP_SQ_thread_id;
2
                                                                                                                         input [6:0] u1_SX_SQ_exp_buf_avail;
       output [TB_DEPTH-1:0] alu_req_q;
                                                                                                                         output
                                                                                                                                           state_change;
       output [STATE_WIDTH-1:0] alu_state_q;
                                                                                                                         output [2:0]
                                                                                                                                           old_state;
       output [STATUS_WIDTH-1:0] alu_status_q;
                                                                                                                                           dealloc_req
                                                                                                                         output [5:0]
                                                                                                                                           dealloc_space;
       input [TB_ADDR_WIDTH-1:0] alu_winner_q;
                                                                                                                                   dealloc_ack;
       input [0:0]
                       alu_winner_ack;
10
                                                                                                                  10
                                                                                                                                           pop_thread;
                                                                                                                         output
11
       input [0:0]
                               ais0_done;
                                                                                                                  11
                                                                                                                         input [6:0]
                                                                                                                                           param_cache_wptr_q;
12
                               ais0_thread_type;
                                                                                                                  12
13
       input [TID_WIDTH-1:0] ais0_thread_id;
                                                                                                                  13
                                                                                                                         output
                                                                                                                                           busy;
14
                                                                                                                  14
15
                               ais1 done;
                                                                                                                  15
       input [0:0]
                                                                                                                                           clk;
                                                                                                                         input
                               ais1_thread_type;
16
       input [0:0]
                                                                                                                  16
                                                                                                                         input
                                                                                                                                           reset
17
       input [TID_WIDTH-1:0] ais1_thread_id;
                                                                                                                  17
18
                                                                                                                  18
19
      input [0:0] u0_SX_SQ_exp_count_rdy;
                                                                                                                  19
20
      input [0:0] u0_SX_SQ_exp_pos_avail;
                                                                                                                  20
                                                                                                                         // -- internal signals --
21
      //tri1 [0:0] u0_SX_SQ_exp_pos_avail;
                                                                                                                  21
22
      input [6:0] u0_SX_SQ_exp_buf_avail;
                                                                                                                  22
23
                                                                                                                  23
                                                                                                                         reg [STATE WIDTH-1:0] tex state q;
24
      input [0:0] u1_SX_SQ_exp_count_rdy;
                                                                                                                  24
                                                                                                                         reg [STATUS_WIDTH-1:0]
25
      input [0:0] u1_SX_SQ_exp_pos_avail;
                                                                                                                         reg [STATE_WIDTH-1:0] alu_state_q;
                                          Page 9 of 49
                                                                                                                                                           Page 10 of 49
                                                                Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                  Ex. 2101 - sq_thread_buff.v
                                                                                                                         wire [STATUS_WIDTH-1:0] status_data_6;
       reg [STATUS_WIDTH-1:0]
                                     alu status q;
                                                                                                                         wire [STATUS_WIDTH-1:0] status_data_7;
2
                                                                                                                         wire [STATUS WIDTH-1:0] status data 8:
      reg [0:0] tp_done_q;
4
       reg [0:0] tp_thread_type_q;
                                                                                                                         wire [STATUS WIDTH-1:0] status data 9:
       reg [TID_WIDTH-1:0] tp_thread_id_q;
                                                                                                                         wire [STATUS WIDTH-1:0] status data 10;
                                                                                                                         wire [STATUS WIDTH-1:0] status data 11;
       reg [TID_WIDTH-1:0] state_head_ptr_q;
                                                                                                                         wire [STATUS_WIDTH-1:0] status_data_12;
       reg [TID_WIDTH-1:0] state_tail_ptr_q;
                                                                                                                         wire [STATUS_WIDTH-1:0] status_data_13;
       reg [TID_WIDTH-1:0] status_tail_ptr_q;
                                                                                                                         wire [STATUS_WIDTH-1:0] status_data_14;
10
                                                                                                                  10
                                                                                                                         wire [STATUS_WIDTH-1:0] status_data_15;
11
       reg [TID_WIDTH:0] full_cnt_q;
                                                                                                                  11
12
                                                                                                                  12
                                                                                                                         reg [STATUS_WIDTH-1:0] tex_status_read_data;
13
       reg [0:0] tex_winner_ack_q;
                                                                                                                         reg [STATUS_WIDTH-1:0] alu_status_read_data;
14
       reg [0:0] alu_winner_ack_q;
15
                                                                                                                  15
                                                                                                                         wire [0:0] cfs_update;
       reg [0:0] alu_winner_ack_q1;
16
17
      reg [0:0] sx_pos_avail;
                                                                                                                         reg [0:0] sx0_exp_count_rdy_q;
18
      reg [6:0] sx buf avail;
                                                                                                                  18
                                                                                                                         reg [0:0] sx0 exp pos avail q;
19
                                                                                                                         reg [6:0] sx0_exp_buf_avail_q;
      wire [STATUS_WIDTH-1:0] status_data_0;
20
                                                                                                                  20
                                                                                                                         reg [0:0] sx1 exp count rdy q;
21
      wire [STATUS WIDTH-1:0] status data 1;
                                                                                                                  21
                                                                                                                         reg [0:0] sx1_exp_pos_avail_q;
22
      wire [STATUS WIDTH-1:0] status data 2:
                                                                                                                  22
                                                                                                                         reg [6:0] sx1_exp_buf_avail_q;
23
      wire [STATUS_WIDTH-1:0] status_data_3;
                                                                                                                  23
24
       wire [STATUS_WIDTH-1:0] status_data_4;
                                                                                                                  24
                                                                                                                         reg [0:0] pos_avail_q;
25
       wire [STATUS_WIDTH-1:0] status_data_5;
                                                                                                                         reg [6:0] buf_avail_q;
                                          Page 11 of 49
                                                                                                                                                           Page 12 of 49
                                                                Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                  Ex. 2101 - sq_thread_buff.v
```

```
// and alu winner ack cannot occur on the same cycle
 2
        reg [1:0] cfs_phase_q;
                                                                                                                                         wire [0:0] winner_ack = tex_winner_ack | alu_winner_ack;
                                                                                                                                         wire [TID_WIDTH-1:0] winner = tex_winner_ack ? tex_winner_q : alu_winner_q;
        // -- combinational logic --
                                                                                                                                         // - buffer full bit is just the MSB of the location counter
        // - qualify the TP and AIS done signals with the vector type (since same done goes to both
                                                                                                                                         wire [0:0] buffer_full_q = full_cnt_q[TID_WIDTH];
11
        // - the ais dones will not occur on the same cycle since they're interleaved
12
                                                                                                                                       // - rtr the ISM ctl pkt data when buffer is not full AND when not loading CFS update data
13
                                                                                                                                        // - CFS updates take priority over ISM loads
        wire [0:0] qual tp done = tp done q & (tp thread type q == thread type strap);
14
                                                                                                                                        // - CFS updates are coordinates by cfs_phase in the ctl flow sequencers
15
        wire [0:0] qual ais0 done = ais0 done & (ais0 thread type == thread type strap);
                                                                                                                                 15
16
        wire [0:0] qual ais1 done = ais1 done & (ais1 thread type == thread type strap);
                                                                                                                                 16
                                                                                                                                         assign tb rtr = ~buffer full q & ~cfs update;
17
                                                                                                                                 17
18
        wire [0:0] qual_ais_done = qual_ais0_done | qual_ais1_done;
                                                                                                                                 18
19
                                                                                                                                 19
                                                                                                                                        // - busy whenever the buffer is not empty (i.e. when full count != 0)
20
        wire [TID WIDTH-1:0] ais thread id = ais0 done? ais0 thread id : ais1 thread id;
                                                                                                                                 20
21
                                                                                                                                 21
                                                                                                                                         wire [0:0] busy = |full cnt q;
22
                                                                                                                                 22
23
        // - mux the thread id/winner down from two sources to one before sending to the status bits
                                                                                                                                 23
24
        // (to update thread valid)
                                                                                                                                 24
                                                                                                                                         // - push a thread when there's a valid transfer from the ISM
        // - note that tex and alu arbiters provide winner_acks on alternate cycles, so tex_winner_ack
                                                                                                                                 25
                                               Page 13 of 49
                                                                                                                                                                                Page 14 of 49
                                                                         Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                          Ex. 2101 - sq_thread_buff.v
        wire [0:0] push_thread = ism_rts & tb_rtr;
                                                                                                                                         wire [0:0] tex rd en = tex winner ack & state read phase;
 2
                                                                                                                                         wire [0:0] alu rd en = alu winner ack & ~state read phase;
                                                                                                                                         wire [0:0] state_rd_en = tex_rd_en | alu_rd_en;
       // - pop a thread when alu status last instr bit is set (and delayed winner ack pulses)
                                                                                                                                         wire [ISM STATE WIDTH-1:0] ism state rd data;
                                                                                                                                         wire [CFS STATE WIDTH-1:0] cfs state rd data;
        wire [0:0] pop thread = alu status q[12] & alu winner ack q1;
                                                                                                                                         wire [STATE_WIDTH-1:0] state_rd_data;
10
        // -- State Mem connections --
                                                                                                                                 10
                                                                                                                                        // - ISM write -
11
                                                                                                                                 11
12
                                                                                                                                        // - state mem writes are shared btwn ISM ctl_pkt loads and CFS updates from 3 CFSs
                                                                                                                                        // (updates have priority), and ALU Instr Seq updates
13
                                                                                                                                         // - the state mem is divided into three parts: ism_state, cfs_state, and alu_state
      /\!/ - state mem reads are shared btwn tex arb and alu arb (tex and alu reqs are mixed in the buffer)
                                                                                                                                        // - ism info is loaded at the tail of the buffer
18
                                                                                                                                        wire [TID_WIDTH-1:0] ism_state_wr_addr = state_tail_ptr_q;
19
       //wire [3:0] state_rd_addr = state_read_phase ? tex_winner_q : alu_winner_q;
                                                                                                                                         wire [0:0] ism_state_wr_en = push_thread;
       // - note: the state mem read address is the thread id from the winning status register
20
                                                                                                                                 20
     \label{eq:winer_thread_id} wire \ [TID\_WIDTH-1:0] \ tex\_winner\_thread\_id = tex\_status\_read\_data[STATUS\_WIDTH-1:STATUS\_WIDTH-TID\_WIDTH];
21
22
                                                                                                                                 21
                                                                                                                                         wire [ISM STATE WIDTH-1:0] ism state wr data =
                                                                                                                                 22
      \label{eq:wine_model} wire \ [TID\_WIDTH-1:0] \ alu\_winner\_thread\_id = alu\_status\_read\_data[STATUS\_WIDTH-1:STATUS\_WIDTH-TID\_WIDTH];
                                                                                                                                 23
                                                                                                                                           ism lod correct. // 144 bits - gets dropped for vertex thread buffer
      wire [TID_WIDTH-1:0] state_rd_addr = state_read_phase ? tex_winner_thread_id alu_winner_thread_id;
                                                                                                                                 24
                                                                                                                                           ism valid bits, // 64 bits - will get moved to alu state
27
                                                                                                                                           ism_gpr_base, // 7 bits
                                               Page 15 of 49
                                                                                                                                                                                Page 16 of 49
                                                                         Ex. 2101 - sq thread buff.v
                                                                                                                                                                                                          Ex. 2101 - sq thread buff.v
```

```
ism_context id // 3 bits
2
        };
                                                                                                                              assign\ cfs\_update = (tcfs\_update\ \&\ (tcfs\_thread\_type == thread\_type\_strap))\ |
                                                                                                                                          (acfs_update & (acfs_thread_type == thread_type_strap)); // asserted on
                                                                                                                            different cfs_phases
      // - CFS write -
                                                                                                                               wire [12:0] cfs_instr_ptr = cfs_state[CFS_STATE_WIDTH-1:CFS_STATE_WIDTH-13];
                                                                                                                             wire [3:0] cfs_exec_ent = cfs_state[CFS_STATE_WIDTH-14:CFS_STATE_WIDTH-17];
      // - cfs mem is written normally by cfs update, but part of cfs mem is initialized by ism load
                                                                                                                        10
                                                                                                                              wire [0:0] cfs export id = cfs state[CFS STATE WIDTH-18];
       // - first mux between the two ALU CFS update inputs
                                                                                                                            wire [6:0] cfs_param_ptr = cfs_state[CFS_STATE_WIDTH-19:CFS_STATE_WIDTH-25];
10
                               acfs_update = acfs0_update | acfs1_update; // these are asserted
11
12
                                                                                                                        13
                                                                                                                                wire [TID_WIDTH-1:0] cfs_thread_id = cfs_status[STATUS_WIDTH-1:STATUS_WIDTH-
                               acfs_thread_type = cfs_phase_q[1] ? acfs1_thread_type :
                                                                                                                              // status[15] is reserved
15
       wire\ [CFS\_STATE\_WIDTH-1:0] \ \ acfs\_state = cfs\_phase\_q[1]\ ?\ acfs1\_state\ : acfs0\_state;
                                                                                                                               wire [0:0] cfs_alu_instr_pending = cfs_status[14];
       wire [0:0] cfs_pulse_sx = cfs_status[13];
18
                                                                                                                               wire [0:0] cfs_last_instr = cfs_status[12];
19
       // - second mux between the tex and alu cfs update info
                                                                                                                               wire [0:0] cfs_pos_allocated = cfs_status[10];
20
       // - need to swap the mux sel polarity since the update is reg'd out of the CFS and thus occurs
                                                                                                                       21
                                                                                                                               wire [1:0] cfs_alloc_type = cfs_status[9:8];
21
       // one cycle after its associated phase
                                                                                                                               wire [3:0] cfs_alloc_size = cfs_status[7:4];
22
                                                                                                                       23
                                                                                                                               wire [0:0] cfs tex read pending = cfs status[3];
       wire [CFS_STATE_WIDTH-1:0] cfs_state = cfs_phase_q[0] ? tcfs_state : acfs_state;
23
                                                                                                                       24
                                                                                                                               wire [0:0] cfs serial = cfs status[2];
24
       wire [STATUS_WIDTH-1:0] cfs_status = cfs_phase_q[0] ? tcfs_status : acfs_status;
                                                                                                                       25
                                                                                                                              wire [0:0] cfs_resource = cfs_status[1];
       //wire [TID_WIDTH-1:0] cfs_thread_id = cfs_phase_q[0] ? tcfs_status[19:16] :
                                                                                                                               wire [0:0] cfs_thread_valid = cfs_status[0];
                                                                                                                       26
27
       //wire [0:0]
                         cfs thread type = cfs phase[0]? tcfs thread type : acfs thread type:
                                           Page 17 of 49
                                                                                                                                                                    Page 18 of 49
                                                                   Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                            Ex. 2101 - sq_thread_buff.v
       wire [TID_WIDTH-1:0] cfs_state_wr_addr = cfs_update ? cfs_thread_id : state_tail_ptr_q;
                                                                                                                                    status_data_0 or status_data_1 or status_data_2 or status_data_3 or
 2
       wire [0:0] cfs state wr en = push thread | cfs update;
                                                                                                                                    status data 4 or status data 5 or status data 6 or status data 7 or
                                                                                                                                    status data 8 or status data 9 or status data 10 or status data 11 or
4
       wire [12:0] instr ptr = cfs update? cfs instr ptr: {ism instr ptr. LO}: // 13 bits
                                                                                                                                    status data 12 or status data 13 or status data 14 or status data 15
       wire [03:0] exec cnt = cfs update ? cfs exec cnt : 4'b0; // 4 bits
                                                                                                                         5
                                                                                                                         6
                                                                                                                                begin
7
       wire [CFS STATE WIDTH-1:0] cfs state wr data =
                                                                                                                         7
                                                                                                                                    case (tex winner q)
8
                                                                                                                         8
                                                                                                                                     4'h0: tex status read data = status data 0;
         instr_ptr, // 13 bits
                                                                                                                                     4'h1: tex_status_read_data = status_data_1;
                                                                                                                                     4'h2: tex_status_read_data = status_data_2;
10
         exec cnt, // 4 bits
                                                                                                                        10
11
         cfs_export_id, // 1 bit
                                                                                                                        11
                                                                                                                                     4'h3: tex_status_read_data = status_data_3;
12
         cfs_param_ptr // 7 bits - gets dropped for pixel thread buffer
                                                                                                                        12
                                                                                                                                     4'h4: tex_status_read_data = status_data_4;
13
                                                                                                                                     4'h5: tex_status_read_data = status_data_5;
14
                                                                                                                                     4'h6: tex_status_read_data = status_data_6;
15
       // --> note that valid bits and predicate bits will go into a separate memory - alu_state
                                                                                                                                     4'h7: tex_status_read_data = status_data_7;
16
                                                                                                                                     4'h8: tex_status_read_data = status_data_8;
17
                                                                                                                                     4'h9: tex_status_read_data = status_data_9;
18
                                                                                                                                     4'ha: tex_status_read_data = status_data_10;
19
      // -- Status Read Data Muxes --
                                                                                                                                     4'hb: tex_status_read_data = status_data_11;
                                                                                                                                     4'hc: tex status read data = status data 12;
20
                                                                                                                        20
21
                                                                                                                        21
                                                                                                                                     4'hd: tex status_read_data = status_data_13;
                                                                                                                        22
       // - need one mux for the tex info, and one for the alu info (and since either can be in any line,
                                                                                                                                     4'he: tex status read data = status data 14:
22
23
                                                                                                                        23
                                                                                                                                     4'hf: tex status read data = status data 15;
24
      // to mux all lines)
                                                                                                                        24
                                                                                                                                     default: tex status read data = {STATUS WIDTH{X}};
25
                                                                                                                                    endcase // case(tex_winner)
       always @(tex_winner_q or
                                           Page 19 of 49
                                                                                                                                                                    Page 20 of 49
                                                                                                                                                                                            Ex. 2101 - sq_thread_buff.v
                                                                   Ex. 2101 - sq thread buff.v
```

```
4'hf: alu status read data = status data 15;
         end // always @ (tex winner or...
2
                                                                                                                                                default: alu\_status\_read\_data = \{STATUS\_WIDTH\{X\}\};
3
       always @(alu winner q or
                                                                                                                                               endcase // case(tex winner)
            status_data_0 or status_data_1 or status_data_2 or status_data_3 or
                                                                                                                                           end // always @ (tex_winner or..
             status\_data\_4 \ or \ status\_data\_5 \ or \ status\_data\_6 \ or \ status\_data\_7 \ or
             status_data_8 or status_data_9 or status_data_10 or status_data_11 or
             status_data_12 or status_data_13 or status_data_14 or status_data_15
                                                                                                                                         // -- Status Write Decoder --
10
                                                                                                                                  10
             case (alu_winner_q)
11
              4'h0: alu_status_read_data = status_data_0;
                                                                                                                                         // - selects status register to write initial status info from ISM
12
              4'h1: alu_status_read_data = status_data_1;
13
              4'h2: alu_status_read_data = status_data_2;
                                                                                                                                  13
                                                                                                                                         reg [15:0] ism_status_sel;
              4'h3: alu_status_read_data = status_data_3;
15
              4'h4: alu_status_read_data = status_data_4;
                                                                                                                                  15
                                                                                                                                          always @( push thread or status tail ptr q )
              4'h5: alu status read data = status data 5:
16
                                                                                                                                  16
17
              4'h6: alu status read data = status data 6:
                                                                                                                                  17
                                                                                                                                               ism status sel = 16'h0:
              4'h7: alu status read data = status data 7;
18
                                                                                                                                  18
                                                                                                                                               if ( push thread )
19
              4'h8: alu status_read_data = status_data_8;
                                                                                                                                  19
20
              4'h9: alu status read data = status data 9:
                                                                                                                                  20
                                                                                                                                                  ism\_status\_sel[status\_tail\_ptr\_q] = HI;
21
                                                                                                                                  21
              4'ha: alu status read data = status data 10;
22
              4'hb: alu_status_read_data = status_data_11;
                                                                                                                                  22
                                                                                                                                           end
23
              4'hc: alu_status_read_data = status_data_12;
                                                                                                                                 23
24
              4'hd: alu_status_read_data = status_data_13;
                                                                                                                                  24
              4'he: alu_status_read_data = status_data_14;
                                                                                                                                  25
                                               Page 21 of 49
                                                                                                                                                                                 Page 22 of 49
                                                                         Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                           Ex. 2101 - sq_thread_buff.v
       // -- Status Winner Decoder --
 2
                                                                                                                                         // state head and tail - works like a ring
        // - selects winning status register so its valid bit can be cleared (once the winning thread is
                                                                                                                                          wire [0:0] inc state tail ptr = push thread;
                                                                                                                                          wire [0:0] inc_state_head_ptr = pop_thread;
       // it becomes invalid until it has finished running its target instructions)
                                                                                                                                   7
                                                                                                                                          always @(posedge clk)
        reg [15:0] winner_status_sel;
                                                                                                                                               if (reset)
                                                                                                                                                                    state\_tail\_ptr\_q <= 0;
10
        always @( winner_ack or winner )
                                                                                                                                               else\ if\ (\ inc\_state\_tail\_ptr\ )\ \ state\_tail\_ptr\_q <= state\_tail\_ptr\_q + 1;
                                                                                                                                  10
11
                                                                                                                                  11
                                                                                                                                                                   state\_tail\_ptr\_q \mathrel{<=} state\_tail\_ptr\_q;
12
             winner_status_sel = 16'h0;
                                                                                                                                  12
13
             if ( winner_ack )
                                                                                                                                  13
14
                                                                                                                                          always @(posedge clk)
15
                winner_status_sel[winner] = HI;
                                                                                                                                  15
                                                                                                                                           begin
16
                                                                                                                                               if ( reset )
                                                                                                                                                                    state_head_ptr_q <= 0;
17
                                                                                                                                              else\ if\ (\ inc\_state\_head\_ptr\ )\ \ state\_head\_ptr\_q <= state\_head\_ptr\_q + 1;
18
                                                                                                                                  18
                                                                                                                                                                   state_head_ptr_q <= state_head_ptr_q;
19
                                                                                                                                           end
20
                                                                                                                                  20
21
       // -- registers --
                                                                                                                                  21
22
                                                                                                                                  22
23
                                                                                                                                  23
                                                                                                                                         // -- Status Regs Tail Ptr --
24
                                                                                                                                  24
25
       // -- State Mem Head and Tail Ptr --
                                                                                                                                  25
                                               Page 23 of 49
                                                                                                                                                                                 Page 24 of 49
                                                                         Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                           Ex. 2101 - sq_thread_buff.v
```

```
// status tail - status regs are shifted forward on a pop, so head ptr is always zero
                                                                                                                            1
                                                                                                                                   begin
 2
                                                                                                                            2
                                                                                                                                      if ( reset )
       // - the reason for shifting the status regs is to keep the ordering fixed so a
                                                                                                                                                          full ent q \le 0;
                                                                                                                                       else if ( inc_state_tail_ptr ) full_cnt_q <= full_cnt_q + 1; // adding a thread
       // simple priority encoder can be used for arbitration (i.e. the request leaving
       // the head of the list is wired directly to the highest priority input of the
                                                                                                                                       else if ( inc_state_head_ptr ) full_cnt_q <= full_cnt_q - 1; // removing a thread
       // arbiter - on a pop, the entire list is shifted up one keeping the highest
                                                                                                                                                        full_ent_q <= full_ent_q;
       // piority thread at the top of the list)
                                                                                                                                    end
       // - so the tail pointer needs to be decremented on a pop, and incremented on a push
       wire [0:0] inc_status_tail_ptr = push_thread;
       wire [0:0] dec_status_tail_ptr = pop_thread;
                                                                                                                                  // -- State Mem Output Registers --
11
12
       always @(posedge clk)
13
                                                                                                                                 // - register state mem read data for texture and alu winner lookup
       begin
14
                                 status_tail_ptr_q <= 0;
15
                                                                                                                          15
            else if (inc status tail ptr) status tail ptr q <= status tail ptr q + 1;
                                                                                                                                  wire [0:0] load tex state = tex winner ack q;
            else if ( dec_status_tail_ptr ) status_tail_ptr_q <= status_tail_ptr_q - 1;
                                                                                                                          16
16
17
                                                                                                                          17
           else
                              status tail ptr q <= status tail ptr q;
                                                                                                                                  always @(posedge clk)
18
                                                                                                                          18
         end
19
                                                                                                                           19
                                                                                                                                       //if ( reset ) tex_state_q <= 0;
20
                                                                                                                           20
                                                                                                                                      if \ (\ load\_tex\_state\ )\ tex\_state\_q <= state\_rd\_data; \\
21
                                                                                                                          21
                                                                                                                                                tex\_state\_q \le tex\_state\_q;
22
       // -- Buffer Location Counter --
                                                                                                                          22
23
                                                                                                                          23
24
                                                                                                                           24
                                                                                                                                  wire [0:0] load_alu_state = alu_winner_ack_q;
       always @(posedge clk)
                                                                                                                           25
                                             Page 25 of 49
                                                                                                                                                                        Page 26 of 49
                                                                     Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                Ex. 2101 - sq_thread_buff.v
       always @(posedge clk)
                                                                                                                                   begin
 2
        begin
                                                                                                                            2
                                                                                                                                      if ( reset )
                                                                                                                                                          alu status q <= 0;
           if (\ load\_alu\_state\ )\ alu\_state\_q <= state\_rd\_data; \\
                                                                                                                                       else\ if\ (\ load\_alu\_status\ )\ alu\_status\_q <= alu\_status\_read\_data;
            else
                     alu\_state\_q \mathrel{<=} alu\_state\_q;
                                                                                                                                       else
                                                                                                                                                         alu status q <= alu status q;
 5
         end
                                                                                                                            5
                                                                                                                                   end
                                                                                                                                  // - register delays for tex winner ack and alu winner ack
       // -- Status Regs Output Registers --
10
                                                                                                                          10 always @(posedge clk)
11
                                                                                                                          11
12
       // - register status reg read data for texture and alu winner lookup
                                                                                                                          12
                                                                                                                                      tex_winner_ack_q <= tex_winner_ack;
13
                                                                                                                                     alu_winner_ack_q <= alu_winner_ack;
14
                                                                                                                                      alu_winner_ack_q1 <= alu_winner_ack_q;
        wire [0:0] load_tex_status = tex_winner_ack_q;
15
                                                                                                                          15
16
       always @(posedge clk)
17
18
           if ( reset )
                          tex status q <= 0;
                                                                                                                                  // - input registers for signals from TPC
19
           else if ( load_tex_status ) tex_status_q <= tex_status_read_data;
20
           else
                            tex status q <= tex status q;
                                                                                                                          20
                                                                                                                                 always @(posedge clk)
21
                                                                                                                          21
        end
22
                                                                                                                                      tp_done_q <= TP_SQ_data_rdy;
                                                                                                                          22
23
       wire [0:0] load_alu_status = alu_winner_ack_q;
                                                                                                                          23
                                                                                                                                      tp_thread_type_q <= TP_SQ_type;
24
                                                                                                                          24
                                                                                                                                      tp_thread_id_q <= TP_SQ_thread_id;
25
       always @(posedge clk)
                                                                                                                           25
                                            Page 27 of 49
                                                                                                                                                                        Page 28 of 49
                                                                     Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                Ex. 2101 - sq_thread_buff.v
```

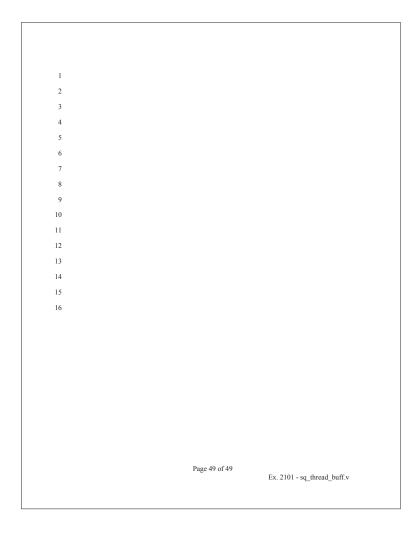
```
begin
 2
                                                                                                                       2
                                                                                                                                sx0\_exp\_count\_rdy\_q \mathrel{<=} u0\_SX\_SQ\_exp\_count\_rdy;
      // - register the output of the state sel muxes
                                                                                                                                 sx0_exp_pos_avail_q <= u0_SX_SQ_exp_pos_avail;
                                                                                                                                  sx0\_exp\_buf\_avail\_q \mathrel{<=} u0\_SX\_SQ\_exp\_buf\_avail;
      // local gfx reg 8:1 muxes (the output is reg'd before use)
                                                                                                                                sx1\_exp\_count\_rdy\_q \mathrel{<=} u1\_SX\_SQ\_exp\_count\_rdy;
                                                                                                                                  sx1\_exp\_pos\_avail\_q \mathrel{<=} u1\_SX\_SQ\_exp\_pos\_avail;
       // --> state needs to come back from the final inst seq SM in order to do the gpr dealloc
                                                                                                                                  sx1\_exp\_buf\_avail\_q \mathrel{<=} u1\_SX\_SQ\_exp\_buf\_avail;
       parameter NUM_REG_WIDTH = 6;
       wire [NUM_REG_WIDTH-1:0] num_reg;
                                                                                                                      10
                                                                                                                             // - save the availability info when valid
11
       reg [NUM_REG_WIDTH-1:0] num_reg_q;
                                                                                                                      11
12
       sq_state_mux #( NUM_REG_WIDTH )
                                                                                                                      12
                                                                                                                            always @(posedge clk)
13
                                                                                                                      13
       num reg sel (.state(3'b0), .input set(num reg set), .mux data out(num reg));
                                                                                                                              begin
14
                                                                                                                                  if ( \ sx0\_exp\_count\_rdy\_q \ | \ sx1\_exp\_count\_rdy\_q \ ) \\
15
                                                                                                                      15
       always @(posedge clk)
16
                                                                                                                      16
                                                                                                                                     pos\_avail\_q \mathrel{<=} sx0\_exp\_pos\_avail\_q \mid sx1\_exp\_pos\_avail\_q;
                                                                                                                           17
           num\_reg\_q \mathrel{<=} num\_reg;
                                                                                                                      18
18
                                                                                                                      19
19
                                                                                                                      20
                                                                                                                              end
20
       wire [5:0] dealloc_space = num_reg_q;
                                                                                                                      21
21
                                                                                                                      22
                                                                                                                             // - need to register cfs_phase since it's used for update data, and update data is reg'd out
22
                                                                                                                      23
23
       // - input register for the SX buffer availability
                                                                                                                      24
                                                                                                                              always @(posedge clk)
24
                                                                                                                      25
25
       always @(posedge clk)
                                                                                                                                  cfs_phase_q <= cfs_phase;
                                           Page 29 of 49
                                                                                                                                                                 Page 30 of 49
                                                                   Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                         Ex. 2101 - sq_thread_buff.v
                                                                                                                               .dealloc_ack(dealloc_ack),
        end
 2
                                                                                                                       2
                                                                                                                               .clk(clk).
                                                                                                                               .reset(reset)
       // -- state machines --
10
       // -- exit state machine --
                                                                                                                      10
                                                                                                                             // -- module instatiations --
11
                                                                                                                     11
12
                                                                                                                      12
13
       // the new state is the current state, which is part of the state of the thread leaving the buffer
14
                                                                                                                            // -- State Memory --
15
       sq_exit_sm
16
       u_sq_exit_sm
17
                                                                                                                      17
                                                                                                                            // - ISM State Mem
18
        .new state rts(LO),
                                                                                                                      18
        .new_state(alu_state_q[CFS_STATE_WIDTH+2:CFS_STATE_WIDTH]),
19
                                                                                                                            dum mem p2
                                                                                                                             #( TB_ADDR_WIDTH, ISM_STATE_WIDTH, TB_DEPTH, 1 )
20
                                                                                                                      20
        //.new state rtr(exit sm rtr),
21
                                                                                                                      21
                                                                                                                             ism state mem
22
        .state diff(state change).
                                                                                                                      22 (
23
                                                                                                                      23
                                                                                                                              .iWEN (ism state wr en),
        .old state q(old state),
24
                                                                                                                      24
                                                                                                                              .iMEW (ism_state_wr_en),
25
        .dealloc_req(dealloc_req),
                                                                                                                               .iWADR (ism_state_wr_addr[3:0]),
                                           Page 31 of 49
                                                                                                                                                                 Page 32 of 49
                                                                  Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                         Ex. 2101 - sq_thread_buff.v
```

```
1
         .iD (ism state wr data),
 2
         .iWCLK (clk).
         .iMER (state_rd_en),
                                                                                                                        4 assign state_rd_data = {ism_state_rd_data, cfs_state_rd_data};
         .iRADR (state_rd_addr[3:0]),
         .oQ(ism_state_rd_data),
         .iRCLK (clk)
                                                                                                                        8 // -- Status Registers --
10
       // - CFS State Mem
11
                                                                                                                       11 sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
                                                                                                                       12 u0_sq_status_reg
12
      dum_mem_p2
13
      #( TB_ADDR_WIDTH, CFS_STATE_WIDTH, TB_DEPTH, 1 )
14
                                                                                                                               .thread type strap(thread type strap),
15
                                                                                                                       15
                                                                                                                                .ism_load (ism_status_sel[0]), // loads ctl pkt info (and sets thread valid status
        .iWEN (cfs_state_wr_en),
16
17
        .iMEW (cfs state wr en).
                                                                                                                       18
                                                                                                                                . ism\_thread\_id \qquad (state\_tail\_ptr\_q), \qquad \textit{// thread id of newly written thread (state tail ptr)}
18
        .iWADR (cfs state wr addr[3:0]),
                                                                                                                       19
                                                                                                                                .ism_resource (ism_resource),
                                                                                                                                                                         // resource bit : tex=1. alu=0
19
         .iD (cfs state wr data),
                                                                                                                               .ism_first_thread (ism_first_thread), // first thread of a new state
20
         .iWCLK (clk),
21
                                                                                                                       22
                                                                                                                                .cfs_update (cfs_update),
                                                                                                                                                                    // load updated status info from CFS
22
        .iMER (state_rd_en),
                                                                                                                       23
                                                                                                                                .cfs_thread_id (cfs_thread_id),
                                                                                                                                                                   // thread ID from CFS
23
        .iRADR (state_rd_addr[3:0]),
                                                                                                                                .cfs_alu_instr_pending(cfs_alu_instr_pending),// alu instr(s) from this thread in alu pipe
24
        .oQ(cfs_state_rd_data),
                                                                                                                                                                         // pulse SX from CFS
                                                                                                                                .cfs_pulse_sx (cfs_pulse_sx),
        .iRCLK (clk)
                                                                                                                                .cfs_last_instr (cfs_last_instr), // last instruction from CFS
                                           Page 33 of 49
                                                                                                                                                                 Page 34 of 49
                                                                   Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                          Ex. 2101 - sq_thread_buff.v
                                                                                                                            .tex_req_q (tex_req_q[0]), tex
         .cfs_pos_allocated(cfs_pos_allocated), // position allocated bit from CFS
                                                                                                                                                              // tex request: simply the thread valid with resource ==
 2
         .cfs_alloc_type (cfs_alloc_type), // alloc type from CFS
                                                                                                                           .alu_req_q (alu_req_q[0]),
availability
                                                                                                                                                             // alu request: function of the status bits and export buffer
         .cfs_alloc_size \, (cfs_alloc_size), \, \, // alloc size from CFS \,
         .cfs tex read pending(cfs tex read pending),// tex pend bit from CFS
         .cfs_serial (cfs_serial), // serial bit from CFS
                                                                                                                               .status\_in\_q \qquad (status\_data\_1), \qquad \textit{// the status input (for shifting)}
                                                                                                                           _status_out_q (status_data_0), // the status output (for shifting, and sending to CFS)
         .cfs_resource (cfs_resource), // resource bit from CFS
         .cfs_thread_valid (cfs_thread_valid), \ \ /\!/ valid bit from CFS
         .sx_pos_avail (pos_avail_q), // position available from SX
                                                                                                                               .reset(reset)
     .sx_buf_avail (buf_avail_q), // buffer available from SX (0 to 127 2kbit buffers(), 2kbit = 32bits for 64 pixels)
                                                                                                                       12
12
         .param_cache_wptr_q (param_cache_wptr_q),
                                                                                                                       13
13
                                                                                                                       14
                                                                                                                              sg status reg #( TID WIDTH, STATUS WIDTH )
14
         //.winner_ack (winner_ack), // winner is selected thread - clears thread_valid
                                                                                                                       15
                                                                                                                              ul sa status reg (
15
         //.winner
                        (winner), //
                                                                                                                       16
                                                                                                                               .thread type strap(thread type strap),
16
         .winner_sel (winner_status_sel[0]), //
                                                                                                                       17
                                                                                                                                .ism_load(ism_status_sel[1]), .ism_thread_id(state_tail_ptr_q),
17
                                                                                                                       18
                                                                                                                                .ism resource(ism resource), .ism first thread(ism first thread),
18
                         (qual_tp_done), // tp done clears tex_read_pending
         .tp done
                                                                                                                       19
                                                                                                                                .cfs update(cfs update),
19
         .tp thread id (tp thread id q), //
                                                                                                                                .cfs_thread_id(cfs_thread_id),
20
                                                                                                                                .cfs_alu_instr_pending(cfs_alu_instr_pending),
21
         ais done
                      (qual ais done),// ais done sets thread valid
                                                                                                                       22
                                                                                                                                .cfs\_pulse\_sx(cfs\_pulse\_sx),
22
         .ais_thread_id (ais_thread_id),//
                                                                                                                                .cfs_last_instr(cfs_last_instr),
23
                                                                                                                                .cfs_pos_allocated(cfs_pos_allocated),
24
         .pop_thread (pop_thread), // this will shift the status regs list
                                                                                                                       25
                                                                                                                                .cfs_alloc_type(cfs_alloc_type),
25
                                                                                                                                .cfs_alloc_size(cfs_alloc_size),
                                           Page 35 of 49
                                                                                                                                                                  Page 36 of 49
                                                                   Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                          Ex. 2101 - sq_thread_buff.v
```

```
.sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
               .cfs tex read pending(cfs tex read pending),
                                                                                                                                                                                                                                                                                                           sx buf avail(buf avail a)
 2
               .cfs serial(cfs serial).
                                                                                                                                                                                                                                      (winner_status_sel[2]),
               .cfs resource(cfs resource).
                                                                                                                                                                                                              .tp\_done(qual\_tp\_done), .tp\_thread\_id(tp\_thread\_id\_q), \\
              .cfs_thread_valid(cfs_thread_valid),
                                                                                                                                                                                                              .ais_done(qual_ais_done), .ais_thread_id(ais_thread_id), .pop_thread(pop_thread),
        .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                           .sx_buf_avail(buf_avail_q),
                                                                                                                                                                                                              .tex\_req\_q(tex\_req\_q[2]), .alu\_req\_q(alu\_req\_q[2]), \\
               .winner_sel (winner_status_sel[1]),
                                                                                                                                                                                                              .status in q(status data 3), .status out q(status data 2),
               .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
                                                                                                                                                                                                              .clk(clk), .reset(reset)
               .ais_done(qual_ais_done), .ais_thread_id(ais_thread_id), .pop_thread(pop_thread),
10
              .tex_req_q(tex_req_q[1]), .alu_req_q(alu_req_q[1]),
11
                                                                                                                                                                                               11
                                                                                                                                                                                                          sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
              .status in q(status data 2), .status out q(status data 1),
12
              .clk(clk), .reset(reset)
                                                                                                                                                                                               12
                                                                                                                                                                                                          u3 sq status reg (
13
                                                                                                                                                                                               13
                                                                                                                                                                                                             .thread type strap(thread type strap),
14
                                                                                                                                                                                               14
                                                                                                                                                                                                             .ism load(ism status sel[3]), .ism thread id(state tail ptr q),
15
            sq status reg #( TID WIDTH, STATUS WIDTH )
                                                                                                                                                                                               15
                                                                                                                                                                                                              .ism resource(ism resource), .ism first thread(ism first thread),
                                                                                                                                                                                                      .cfs_update(cfs_update), .cfs_serial(cfs_serial), .cfs_resource(cfs_resource), .cfs_alloe_type(cfs_alloe_type),
16
            u2 sa status reg (
17
               .thread_type_strap(thread_type_strap),
                                                                                                                                                                                                      .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                                                                                                      .cfs_pos_allocated(cfs_pos_allocated),
18
               .ism_load(ism_status_sel[2]), .ism_thread_id(state_tail_ptr_q),
19
               . ism\_resource (ism\_resource), . ism\_first\_thread (ism\_first\_thread), \\
                                                                                                                                                                                                       .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                                                                                                                                                                                              .cfs_thread_valid(cfs_thread_valid),
         .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                         .cfs_serial(cfs_serial), .cfs_resource(cfs_resource),
21
                                                                                                                                                                                                             .cfs_thread_id(cfs_thread_id), .cfs_alu_instr_pending(cfs_alu_instr_pending),
         .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                       .cfs_pos_allocated(cfs_pos_allocated),
                                                                                                                                                                                                      .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
22
23
                                                                                                                                                                                                                                                                                                           .sx_buf_avail(buf_avail_q),
24
25
        .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                              .cfs_thread_valid(cfs_thread_valid),
                                                                                                                                                                                               25
                                                                                                                                                                                                            .winner sel (winner status sel[3]),
                                                                                                                                                                                               26
                                                                                                                                                                                                            .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
              .cfs\_thread\_id(cfs\_thread\_id), .cfs\_alu\_instr\_pending(cfs\_alu\_instr\_pending),\\
                                                                                                                                                                                                             .ais_done(qual_ais_done), .ais_thread_id(ais_thread_id), .pop_thread(pop_thread),
                                                                      Page 37 of 49
                                                                                                                                                                                                                                                                     Page 38 of 49
                                                                                                            Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                                                                                                                           Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                         sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
               .tex req q(tex req q[3]), .alu req q(alu req q[3]),
 2
                                                                                                                                                                                                 2 u5 sq status reg (
              .status in q(status data 4), .status out q(status data 3),
 3
               .clk(clk), .reset(reset)
                                                                                                                                                                                                             .thread type strap(thread type strap),
 4
                                                                                                                                                                                                             .ism load(ism status sel[5]), .ism thread id(state tail ptr q),
 5
                                                                                                                                                                                                              .ism_resource(ism_resource), .ism_first_thread(ism_first_thread),
                                                                                                                                                                                                       .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                                                                                                                                                                                                                         .cfs_serial(cfs_serial), .cfs_resource(cfs_resource),
          sq status reg #( TID WIDTH, STATUS WIDTH )
           u4_sq_status_reg (
                                                                                                                                                                                                      .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                                                                                                      .cfs_pos_allocated(cfs_pos_allocated),
             . thread\_type\_strap (thread\_type\_strap),
              . ism\_load(ism\_status\_sel[4]), . ism\_thread\_id(state\_tail\_ptr\_q), \\
                                                                                                                                                                                                      .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                                                                                                                                                                                             .cfs_thread_valid(cfs_thread_valid),
              . ism\_resource (ism\_resource), . ism\_first\_thread (ism\_first\_thread), \\
10
                                                                                                                                                                                               12
                                                                                                                                                                                                             .cfs thread id(cfs thread id), .cfs alu instr pending(cfs alu instr pending),
         .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                         .cfs_serial(cfs_serial), .cfs_resource(cfs_resource),
                                                                                                                                                                                                        .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                                                                                                                                                                                                                           .sx buf avail(buf avail q),
         .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                        .cfs_pos_allocated(cfs_pos_allocated),
                                                                                                                                                                                               15
                                                                                                                                                                                                            .winner_sel (winner_status_sel[5]),
               .cfs pulse sx(cfs pulse sx),
                                                                                            .cfs thread valid(cfs thread valid),
                                                                                                                                                                                               16
                                                                                                                                                                                                              .tp\_done(qual\_tp\_done), .tp\_thread\_id(tp\_thread\_id\_q), \\
 16
         .cfs tex read pending(cfs tex read pending),
                                                                                                                                                                                               17
                                                                                                                                                                                                             . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), .pop\_thr
17
              .cfs_thread_id(cfs_thread_id), .cfs_alu_instr_pending(cfs_alu_instr_pending),
                                                                                                                                                                                               18
                                                                                                                                                                                                             .tex_req_q(tex_req_q[5]), .alu_req_q(alu_req_q[5]),
         .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                          .sx buf avail(buf avail q),
                                                                                                                                                                                               19
                                                                                                                                                                                                             .status_in_q(status_data_6), .status_out_q(status_data_5),
20
               .winner_sel (winner_status_sel[4]),
                                                                                                                                                                                               20
                                                                                                                                                                                                             .clk(clk), .reset(reset)
21
               .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
                                                                                                                                                                                               21
22
               .ais_done(qual_ais_done), .ais_thread_id(ais_thread_id), .pop_thread(pop_thread),
                                                                                                                                                                                               22
23
               .tex_req_q(tex_req_q[4]), .alu_req_q(alu_req_q[4]),
                                                                                                                                                                                               23
                                                                                                                                                                                                           sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
24
               .status\_in\_q(status\_data\_5), .status\_out\_q(status\_data\_4), \\
                                                                                                                                                                                               24
                                                                                                                                                                                                           u6_sq_status_reg (
25
                                                                                                                                                                                               25
                                                                                                                                                                                                             .thread\_type\_strap(thread\_type\_strap),
26
            );
                                                                                                                                                                                                             .ism_load(ism_status_sel[6]), .ism_thread_id(state_tail_ptr_q)
                                                                                                                                                                                                              . ism\_resource (ism\_resource), . ism\_first\_thread (ism\_first\_thread), \\
                                                                      Page 39 of 49
                                                                                                                                                                                                                                                                     Page 40 of 49
                                                                                                            Ex. 2101 - sq thread buff.v
                                                                                                                                                                                                                                                                                                           Ex. 2101 - sq thread buff.v
```

```
.cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                                                .cfs serial(cfs serial),
                                                                                                                                         .cfs resource(cfs resource).
                                                                                                                                                                                                                                                                     .cfs thread id(cfs thread id), .cfs alu instr pending(cfs alu instr pending),
                                                                                                                                                                                                                                                           .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                                                                                                                                                                                                                                                                                                          .sx buf avail(buf avail q),
            .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                                                    .cfs_pos_allocated(cfs_pos_allocated),
                                                                                                                                                                                                                                                                    .winner_sel (winner_status_sel[7]),
             .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                     .cfs thread valid(cfs thread valid).
                                                                                                                                                                                                                                                                     .tp\_done(qual\_tp\_done), .tp\_thread\_id(tp\_thread\_id\_q), \\
                                                                                                                                                                                                                                                                      . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), .pop\_thread(pop\_thread(pop\_thread), .pop\_thread(pop\_thread), .pop\_thread(pop\_thread
                   .cfs thread id(cfs thread id), .cfs alu instr pending(cfs alu instr pending),
                    .sx pos avail(pos avail q),
                                                                                                                                       .sx buf avail(buf avail q),
                                                                                                                                                                                                                                                                      .tex\_req\_q(tex\_req\_q[7]), \ .alu\_req\_q(alu\_req\_q[7]), \\
             .param_cache_wptr_q(param_cache_wptr_q),
                                                                                                                                                                                                                                                                      .status_in_q(status_data_8), .status_out_q(status_data_7),
10
                   winner sel (winner status sel[6])
                                                                                                                                                                                                                                                                      .clk(clk), .reset(reset)
11
                    .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
12
                   . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), \\
                                                                                                                                                                                                                                                  11
13
                    .tex_req_q(tex_req_q[6]), .alu_req_q(alu_req_q[6]),
                                                                                                                                                                                                                                                   12
                                                                                                                                                                                                                                                                 sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
14
                   .status\_in\_q(status\_data\_7), .status\_out\_q(status\_data\_6), \\
                                                                                                                                                                                                                                                  13
                                                                                                                                                                                                                                                                 u8 sq status reg (
                   clk(clk), .reset(reset)
15
                                                                                                                                                                                                                                                  14
                                                                                                                                                                                                                                                                   .thread type strap(thread type strap),
16
                                                                                                                                                                                                                                                  15
                                                                                                                                                                                                                                                                      .ism load(ism status sel[8]), .ism thread id(state tail ptr q),
17
                                                                                                                                                                                                                                                  16
                                                                                                                                                                                                                                                                      .ism resource(ism resource), .ism first thread(ism first thread),
18
               sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
                                                                                                                                                                                                                                                           .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                                                                                                                                                                                                                                                                                              .cfs_serial(cfs_serial),
                                                                                                                                                                                                                                                                                                                                                                                          .cfs_resource(cfs_resource),
19
               u7_sq_status_reg (
                                                                                                                                                                                                                                                              .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                                                                                                                                                                                     .cfs_pos_allocated(cfs_pos_allocated),
20
                   .thread_type_strap(thread_type_strap),
21
                   .ism_load(ism_status_sel[7]), .ism_thread_id(state_tail_ptr_q),
                                                                                                                                                                                                                                                                                                                                                                         .cfs_thread_valid(cfs_thread_valid),
                                                                                                                                                                                                                                                                      .cfs_pulse_sx(cfs_pulse_sx),
22
                                                                                                                                                                                                                                                               .cfs_tex_read_pending(cfs_tex_read_pending),
                   .ism resource(ism resource), .ism first thread(ism first thread),
           .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
23
24
                                                                            .cfs_serial(cfs_serial),
                                                                                                                                                                                                                                                                     .cfs_thread_id(cfs_thread_id), .cfs_alu_instr_pending(cfs_alu_instr_pending),
                                                                                                                                       .cfs resource(cfs resource),
                                                                                                                                                                                                                                                          .sx\_pos\_avail(pos\_avail\_q), \\ .param\_cache\_wptr\_q(param\_cache\_wptr\_q), \\
                                                                                                                                                                                                                                                                                                                                                                                           .sx buf avail(buf avail q),
25
                    .cfs alloc size(cfs alloc size).
                                                                                                                   .cfs pos allocated(cfs pos allocated),
26
           .cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                                                                  26
                                                                                                                                                                                                                                                                   .winner_sel (winner_status_sel[8]),
           .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                          .cfs thread valid(cfs thread valid),
                                                                                                                                                                                                                                                                    .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
                                                                                                                                                                                                                                                                                                                                           Page 42 of 49
                                                                                         Page 41 of 49
                                                                                                                                                                                                                                                                                                                                                                                           Ex. 2101 - sq_thread_buff.v
                                                                                                                                         Ex. 2101 - sq_thread_buff.v
                   .ais_done(qual_ais_done), .ais_thread_id(ais_thread_id), .pop_thread(pop_thread),
  2
                                                                                                                                                                                                                                                    2
                                                                                                                                                                                                                                                                 sq status reg #( TID WIDTH, STATUS WIDTH )
                   .tex req q(tex req q[8]), .alu req q(alu req q[8]),
                   .status in q(status data 9), .status out q(status data 8),
                                                                                                                                                                                                                                                                 u10 sq status reg (
                   .clk(clk). .reset(reset)
                                                                                                                                                                                                                                                                    .thread type strap(thread type strap),
  5
                                                                                                                                                                                                                                                                     .ism_load(ism_status_sel[10]), .ism_thread_id(state_tail_ptr_q),
                                                                                                                                                                                                                                                                     .ism_resource(ism_resource), .ism_first_thread(ism_first_thread),
                                                                                                                                                                                                                                                              .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                                                                                                                                                                                                                                                                                            .cfs_serial(cfs_serial),
 7
              sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
                                                                                                                                                                                                                                                                                                                                                                                          .cfs resource(cfs resource),
 8
              u9_sq_status_reg (
                                                                                                                                                                                                                                                              .cfs_alloc_size(cfs_alloc_size), .cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                                                                                                                                                                                    .cfs_pos_allocated(cfs_pos_allocated),
                  .thread_type_strap(thread_type_strap),
10
                  . ism\_load(ism\_status\_sel[9]), . ism\_thread\_id(state\_tail\_ptr\_q), \\
                                                                                                                                                                                                                                                                      .cfs_pulse_sx(cfs_pulse_sx),
                                                                                                                                                                                                                                                                                                                                                                         .cfs thread valid(cfs thread valid),
                                                                                                                                                                                                                                                              .cfs tex read pending(cfs tex read pending),
11
                   .ism_resource(ism_resource), .ism_first_thread(ism_first_thread),
                                                                                                                                                                                                                                                   13
                                                                                                                                                                                                                                                                    .cfs thread id(cfs thread id), .cfs alu instr pending(cfs alu instr pending),
           .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                                        .cfs_serial(cfs_serial), .cfs_resource(cfs_resource),
                                                                                                                                                                                                                                                              .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                                                                                                                                                                                                                                                                                                         .sx buf avail(buf avail q).
                                                                                                                                                                                                                                                   15
14
15
                    .cfs_alloc_size(cfs_alloc_size),
                                                                                                               .cfs_pos_allocated(cfs_pos_allocated),
             .cfs last instr(cfs_last_instr),
                                                                                                                                                                                                                                                  16
                                                                                                                                                                                                                                                                    .winner sel (winner status sel[10]),
           .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
16
17
                                                                                                                       .cfs thread valid(cfs thread valid),
                                                                                                                                                                                                                                                  17
                                                                                                                                                                                                                                                                      .tp\_done(qual\_tp\_done), \ .tp\_thread\_id(tp\_thread\_id\_q), \\
                                                                                                                                                                                                                                                  18
                                                                                                                                                                                                                                                                     . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), .pop\_thr
18
                  .cfs\_thread\_id(cfs\_thread\_id), .cfs\_alu\_instr\_pending(cfs\_alu\_instr\_pending), \\
                                                                                                                                                                                                                                                  19
                                                                                                                                                                                                                                                                      .tex\_req\_q(tex\_req\_q[10]), .alu\_req\_q(alu\_req\_q[10]), \\
             .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                                                        .sx buf avail(buf avail q),
                                                                                                                                                                                                                                                  20
                                                                                                                                                                                                                                                                      .status_in_q(status_data_11), .status_out_q(status_data_10),
21
                   .winner_sel (winner_status_sel[9]),
                                                                                                                                                                                                                                                  21
                                                                                                                                                                                                                                                                     .clk(clk), .reset(reset)
22
                                                                                                                                                                                                                                                  22
                   .tp\_done(qual\_tp\_done), .tp\_thread\_id(tp\_thread\_id\_q), \\
23
                    .ais_done(qual_ais_done), .ais_thread_id(ais_thread_id), .pop_thread(pop_thread),
                                                                                                                                                                                                                                                  23
                                                                                                                                                                                                                                                                 sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
24
                   . tex\_req\_q(tex\_req\_q[9]), \ . alu\_req\_q(alu\_req\_q[9]), \\
                                                                                                                                                                                                                                                  24
25
                   .status\_in\_q(status\_data\_10), .status\_out\_q(status\_data\_9), \\
                                                                                                                                                                                                                                                  25
                                                                                                                                                                                                                                                                 ull sq status reg (
                  .clk(clk), .reset(reset)
26
                                                                                                                                                                                                                                                                     .thread_type_strap(thread_type_strap),
                                                                                                                                                                                                                                                                      .ism_load(ism_status_sel[11]), .ism_thread_id(state_tail_ptr_q),
                                                                                         Page 43 of 49
                                                                                                                                                                                                                                                                                                                                           Page 44 of 49
                                                                                                                                         Ex. 2101 - sq thread buff.v
                                                                                                                                                                                                                                                                                                                                                                                           Ex. 2101 - sq thread buff.v
```

```
.cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
               .ism_resource(ism_resource), .ism_first_thread(ism_first_thread),
                                                                                                                                                                                                                                                                                                 .cfs thread valid(cfs thread valid),
         .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                              .cfs_serial(cfs_serial),
                                                                                                           .cfs resource(cfs resource),
                                                                                                                                                                                                              .cfs_thread_id(cfs_thread_id), .cfs_alu_instr_pending(cfs_alu_instr_pending),
         .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                       .cfs_pos_allocated(cfs_pos_allocated),
                                                                                                                                                                                                          .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
                                                                                                                                                                                                                                                                                                             .sx buf avail(buf avail q),
         .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                              .cfs_thread_valid(cfs_thread_valid),
                                                                                                                                                                                                             .winner_sel (winner_status_sel[12]),
                                                                                                                                                                                                              .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
              .cfs thread id(cfs thread id), .cfs alu instr pending(cfs alu instr pending),
                                                                                                                                                                                                              .ais done(qual ais done), .ais thread id(ais thread id), .pop thread(pop thread),
                sx pos avail(pos avail q).
                                                                                                            .sx buf avail(buf avail q).
                                                                                                                                                                                                               .tex_req_q(tex_req_q[12]), .alu_req_q(alu_req_q[12]),
10
         .param_cache_wptr_q(param_cache_wptr_q),
               .winner_sel (winner_status_sel[11]),
                                                                                                                                                                                                              .status in q(status data 13), .status out q(status data 12),
11
                                                                                                                                                                                                              .clk(clk), .reset(reset)
12
               .tp\_done(qual\_tp\_done), \ .tp\_thread\_id(tp\_thread\_id\_q), \\
                                                                                                                                                                                                12
13
               . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), \\
                                                                                                                                                                                                13
14
               .tex\_req\_q(tex\_req\_q[11]), .alu\_req\_q(alu\_req\_q[11]), \\
                                                                                                                                                                                                            sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
15
               .status_in_q(status_data_12), .status_out_q(status_data_11),
                                                                                                                                                                                                14
16
               clk(clk), .reset(reset)
                                                                                                                                                                                                15
                                                                                                                                                                                                            u13 sa status reg (
17
                                                                                                                                                                                                16
                                                                                                                                                                                                              .thread_type_strap(thread_type_strap),
18
                                                                                                                                                                                                17
                                                                                                                                                                                                              . ism\_load(ism\_status\_sel[13]), . ism\_thread\_id(state\_tail\_ptr\_q), \\
            sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
                                                                                                                                                                                                18
                                                                                                                                                                                                              . ism\_resource (ism\_resource), . ism\_first\_thread (ism\_first\_thread), \\
19
                                                                                                                                                                                                       .cfs_update(cfs_update), .cfs_serial(cfs_serial), .cfs_alloc_type(cfs_alloc_type),
                                                                                                                                                                                                                                                                                                            .cfs_resource(cfs_resource),
20
            u12_sq_status_reg (
21
               .thread_type_strap(thread_type_strap),
                                                                                                                                                                                                                                                                                           .cfs_pos_allocated(cfs_pos_allocated),
                                                                                                                                                                                                               .cfs_alloc_size(cfs_alloc_size),
22
                                                                                                                                                                                                        .cfs_last_instr(cfs_last_instr),
               .ism load(ism status sel[12]), .ism thread id(state tail ptr q),
23
               .ism resource(ism resource), .ism first thread(ism first thread),
                                                                                                                                                                                                       .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                                                                                                                                                                                                .cfs_thread_valid(cfs_thread_valid),
       .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                             .cfs serial(cfs serial),
24
25
                                                                                                      .cfs resource(cfs resource),
                                                                                                                                                                                                25
                                                                                                                                                                                                              .cfs\_thread\_id(cfs\_thread\_id), .cfs\_alu\_instr\_pending(cfs\_alu\_instr\_pending), \\
         .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                          .sx\_pos\_avail(pos\_avail\_q),\\.param\_cache\_wptr\_q(param\_cache\_wptr\_q),
                                                                                       .cfs pos allocated(cfs pos allocated),
                                                                                                                                                                                                                                                                                                             .sx buf avail(buf avail q),
                                                                                                                                                                                                             .winner_sel (winner_status_sel[13]),
                                                                      Page 45 of 49
                                                                                                                                                                                                                                                                      Page 46 of 49
                                                                                                            Ex. 2101 - sq_thread_buff.v
                                                                                                                                                                                                                                                                                                             Ex. 2101 - sq_thread_buff.v
               .tp done(qual tp done), .tp thread id(tp thread id q),
  2
                                                                                                                                                                                                  2
               .ais done(qual ais done), .ais thread id(ais thread id), .pop thread(pop thread),
                                                                                                                                                                                                           sq status reg #( TID WIDTH, STATUS WIDTH )
               .tex\_req\_q(tex\_req\_q[13]), .alu\_req\_q(alu\_req\_q[13]), \\
               .status in q(status data 14), .status out q(status data 13),
                                                                                                                                                                                                           u15 sq status reg (
  5
               clk(clk) reset(reset)
                                                                                                                                                                                                              .thread_type_strap(thread_type_strap),
  6
                                                                                                                                                                                                              .ism_load(ism_status_sel[15]), .ism_thread_id(state_tail_ptr_q),
                                                                                                                                                                                                               . ism\_resource (ism\_resource), . ism\_first\_thread (ism\_first\_thread), \\
            sq_status_reg #( TID_WIDTH, STATUS_WIDTH )
                                                                                                                                                                                                         .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                                                                                                                                                                                                                          .cfs_serial(cfs_serial), .cfs_resource(cfs_resource),
 8
           u14_sq_status_reg (
                                                                                                                                                                                                         .cfs_alloc_size(cfs_alloc_size),
.cfs_last_instr(cfs_last_instr),
                                                                                                                                                                                                                                                                                         .cfs_pos_allocated(cfs_pos_allocated),
10
             .thread_type_strap(thread_type_strap),
11
               .ism_load(ism_status_sel[14]), .ism_thread_id(state_tail_ptr_q),
                                                                                                                                                                                                       .cfs_pulse_sx(cfs_pulse_sx),
.cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                                                                                                                                                                                                .cfs_thread_valid(cfs_thread_valid),
12
               .ism_resource(ism_resource), .ism_first_thread(ism_first_thread),
                                                                                                                                                                                                14
                                                                                                                                                                                                              .cfs_thread_id(cfs_thread_id), .cfs_alu_instr_pending(cfs_alu_instr_pending),
         .cfs_update(cfs_update),
.cfs_alloc_type(cfs_alloc_type),
                                                         .cfs_serial(cfs_serial), .cfs_resource(cfs_resource),
                                                                                                                                                                                                               .sx_pos_avail(pos avail q),
                                                                                                                                                                                                                                                                                                             .sx buf avail(buf avail q).
                                                                                                                                                                                                 16
                                                                                                                                                                                                          .param_cache_wptr_q(param_cache_wptr_q),
         .cfs_alloc_size(cfs_alloc_size),
.cfs last instr(cfs last instr),
                                                                                         .cfs_pos_allocated(cfs_pos_allocated),
                                                                                                                                                                                                17
                                                                                                                                                                                                             .winner sel (winner status sel[15]),
               .cfs pulse_sx(cfs_pulse_sx),
 17
18
                                                                                             .cfs thread valid(cfs thread valid),
                                                                                                                                                                                                18
                                                                                                                                                                                                               .tp\_done(qual\_tp\_done), .tp\_thread\_id(tp\_thread\_id\_q), \\
         .cfs_tex_read_pending(cfs_tex_read_pending),
                                                                                                                                                                                                19
                                                                                                                                                                                                               . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), .pop\_thr
19
              .cfs\_thread\_id(cfs\_thread\_id), .cfs\_alu\_instr\_pending(cfs\_alu\_instr\_pending), \\
                                                                                                                                                                                                20
                                                                                                                                                                                                               .tex\_req\_q(tex\_req\_q[15]), .alu\_req\_q(alu\_req\_q[15]), \\
         .sx_pos_avail(pos_avail_q),
.param_cache_wptr_q(param_cache_wptr_q),
20
21
                                                                                                          .sx_buf_avail(buf_avail_q),
                                                                                                                                                                                                21
                                                                                                                                                                                                               .status\_in\_q(\{STATUS\_WIDTH\{LO\}\}), .status\_out\_q(status\_data\_15),\\
22
                                                                                                                                                                                                22
               .winner_sel (winner_status_sel[14]),
23
               .tp_done(qual_tp_done), .tp_thread_id(tp_thread_id_q),
                                                                                                                                                                                                23
24
                                                                                                                                                                                                24
               . a is\_done(qual\_ais\_done), . a is\_thread\_id(ais\_thread\_id), .pop\_thread(pop\_thread), \\
25
                                                                                                                                                                                                25
               . tex\_req\_q(tex\_req\_q[14]), . alu\_req\_q(alu\_req\_q[14]), \\
               .status\_in\_q(status\_data\_15), .status\_out\_q(status\_data\_14), \\
                                                                                                                                                                                                         endmodule
                                                                     Page 47 of 49
                                                                                                                                                                                                                                                                      Page 48 of 49
                                                                                                            Ex. 2101 - sq thread buff.v
                                                                                                                                                                                                                                                                                                             Ex. 2101 - sq thread buff.v
```



```
1 'include "header.v"
                                                                                                                        1 // an instruction to the IO
                                                                                                                        2 //
 4 \hspace{0.5cm} // \hspace{0.1cm} \$ Id: // depot/r400/ devel/parts\_lib/src/gfx/sq/tis/sq\_target\_instr\_fetch.v\#15 \hspace{0.1cm} \$
                                                                                                                        4 //-
                                                                                                                        7 'include "sq_defs.v"
 8 // Copyright: Trade secret of ATI Technologies, Inc.
                   © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
                                                                                                                        9 module sq_target_instr_fetch
10 //
                                                                                                                       10 (
11 //
                 All rights reserved. This notice is intended as a precaution against
                                                                                                                       11 target_strap, // hardwired to TEX_STRAP, ALU0_STRAP, or ALU1_STRAP
                 inadvertent publication and does not imply publication or any waiver
                                                                                                                       12
                                                                                                                                                // (instr store phase compared to strap)
13 //
                 of confidentiality. The year included in the foregoing notice is the
                                                                                                                       13
14 //
                year of creation of the work.
                                                                                                                       14 // local registers
15 //
                                                                                                                       15 // - per chip
17 inst_base_pix,
                                                                                                                                                // pixel base
                                                                                                                       18 // - per context
18 // target instr fetch.v
20 // - loads the initial instr_ptr into the TIP register (target instruction pointer)
                                                                                                                       20 // cfs interface
21 // - loads count input into TIC (target instruction counter)
                                                                                                                       21 cfs rts,
                                                                                                                                                 // ctl packet and ptr are valid
22 // - reads instructions into TIR (target instruction register)
                                                                                                                       22
                                                                                                                             cfs_ctl_pkt,
                                                                                                                                                 // the control packet (lod for pix_tex, valid_bits, gpr_base, context_id)
23 // - outputs TIR to decode pipe
                                                                                                                       23 cfs export info, //
24 // - loads other state info into an input staging register
                                                                                                                       24 cfs_instr_ptr,
                                                                                                                                               // the Instruction Store address of the first target instruction
25 // - transfers input staging register to an output staging register when ready to send
                                                                                                                       25 cfs_instr_cnt,
                                                                                                                                               // the number of instructions to be fetched
                                            Page 1 of 20
                                                                                                                                                                   Page 2 of 20
                                                             Ex. 2102 - sq target instr fetch.v
                                                                                                                                                                                     Ex. 2102 - sq target instr fetch.v
 1 cfs_pc_base, // the param cache base (alloc'd in arbiter)
                                                                                                                              clk.
                                                                                                                        2
 2 cfs_thread_type, // vertex or pixel
                                                                                                                              reset
       cfs_thread_id,
       cfs_last_in_thread, // last_instr status bit
 5 tif rtr,
                       // TIF can take a new packet
                                                                                                                             // -- parameters --
 7 // instruction store interface
                                                                                                                             parameter CTL_PKT_WIDTH = 8; // number of bits in the control packet (drop the lod_correct bits on ALU side)
 8 is_read_addr, // instruction store read address
       is_read_data, // instruction store read data
                                                                                                                             parameter VTX = 'SQ_VTX;
                                                                                                                       11
                                                                                                                              parameter PIX = `SQ_PIX;
11
      alu_phase,
                       // alu phase (alu0 and alu1 share the alu is_phase)
                                                                                                                       12
                                                                                                                       13 parameter LO = 1'b0;
13 // outputs to the target instruction decoder (in the TIQ module)
                                                                                                                       14 parameter HI = 1'b1;
14 tif_pc_base_q, // the param cache base output
                                                                                                                       15
                                                                                                                             parameter X = 1'bx;
15
                         // the target control packet (pipelined from reg'd input)
      tif ctl pkt q,
                                                                                                                       16
       tif_export_info_q, // the target control packet (pipelined from reg'd input)
                                                                                                                       17
17 tif_last_in_group_q,// last instruction in series of consecutive tex/alu instructions
                                                                                                                       18
18 tif last in thread q,// last instruction in thread
                                                                                                                       19
19 tif_thread_type_q, // vert:1, pix:0
20 \qquad tif\_thread\_id\_q, \qquad \textit{// the target thread id}
                                                                                                                       21
                         // the target instruction register (TIR)
21 tif instr q.
                                                                                                                       22
                                                                                                                             input [2:0] target strap;
       tif_instr_rts_q, // the target instr register is valid
23
                          // the target instr decode is ready to take the TIR (and other pipeline data)
                                                                                                                       24
                                                                                                                              input [11:0] inst_base_vtx;
24
                                                                                                                       25
                                                                                                                             input [11:0] inst base pix;
                                            Page 3 of 20
                                                                                                                                                                   Page 4 of 20
                                                              Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                     Ex. 2102 - sq_target_instr_fetch.v
```

ATI 2102 LG v. ATI IPR2015-00326

```
input
                          cfs rts;
                                                                                                                               output [6:0]
                                                                                                                                                  tif_pc_base_q;
 2
       input [11:0]
                          cfs_instr_ptr;
                                                                                                                               reg [6:0]
                                                                                                                                                        tif_pc_base_q;
       input [11:0]
                          cfs instr cnt;
                                                                                                                               output
                                                                                                                                                        tif_instr_rts_q;
       input [CTL_PKT_WIDTH-1:0] cfs_ctl_pkt;
                                                                                                                                                        tif_instr_rts_q;
       input [5:0]
                               cfs thread id;
                                                                                                                                                         tif_last_in_group_q;
       input [1:0]
                                cfs_export_info;
                                                                                                                                                         tif_last_in_group_q;
                                cfs_pc_base;
                                                                                                                                                         tif_last_in_thread_q;
                         cfs_thread_type;
                                                                                                                                                         tif_last_in_thread_q;
                         cfs_last_in_thread;
                                                                                                                                                         tif_thread_type_q;
       input
                                                                                                                               output
                               tif_rtr;
                                                                                                                                                         tif_thread_type_q;
       output
11
                                 tif_rtr;
                                                                                                                        11
12
                                                                                                                        12
                                                                                                                               input
                                                                                                                                                  tiq_rtr;
13
       output [11:0]
                          is_read_addr;
                                                                                                                        13
14
       input [95:0] is_read_data;
                                                                                                                               output busy;
15
       input [1:0]
                                                                                                                        15
                          is phase;
                                                                                                                               input clk;
16
                                                                                                                        16
       input [0:0]
                          alu_phase;
                                                                                                                               input reset;
17
                                                                                                                        17
                                                                                                                        18
18
       output [95:0]
                          tif instr a:
19
       reg [95:0]
                          tif instr q;
                                                                                                                        19
       output [CTL_PKT_WIDTH-1:0] tif_ctl_pkt_q;
20
                                                                                                                               // -- internal signals --
                                                                                                                        20
21
       reg [CTL_PKT_WIDTH-1:0] tif_ctl_pkt_q;
                                                                                                                       21
22
       output [5:0]
                          tif thread id q;
                                                                                                                       22
23
       reg [5:0]
                               tif_thread_id_q;
                                                                                                                       23
                                                                                                                               reg [11:0] tic_q;
24
       output [1:0]
                          tif_export_info_q;
                                                                                                                       24
                                                                                                                               reg [11:0] tip_q;
       reg [1:0]
                               tif_export_info_q;
                                                                                                                        25
                                            Page 5 of 20
                                                                                                                                                                    Page 6 of 20
                                                              Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                      Ex. 2102 - sq_target_instr_fetch.v
       reg [CTL_PKT_WIDTH-1:0] isr_ctl_pkt_q;
       reg [1:0]
                              isr export info q;
                                 isr_pc_base_q;
       reg [6:0]
                                                                                                                               // ld tir is a SM reg'd out signal
       reg [5:0]
                                 isr thread id q;
                                                                                                                               wire busy = |current_state | ld_tir | tif_instr_rts_q | tif_instr_rts_q1;
       reg
                                  isr_thread_type_q;
                                  isr_last_in_thread_q;
       reg
                                                                                                                              // - just re-assign the IS read address to the TIP
       reg ld tir;
                                                                                                                               assign is read addr = tip q;
       reg ld tip;
       reg inc_tip;
10
                                                                                                                        10
11
       reg dec_tic;
                                                                                                                        11
12
                                                                                                                        12
13
14
       reg [1:0] current_state;
15
       reg [1:0] next_state;
                                                                                                                               // -- Target Instruction Pointer (TIP) --
17
       reg tif_instr_rts_q1;
18
                                                                                                                              // - initially loaded with instr_ptr from CFS
19
                                                                                                                               // - inc'd by 1 to next sequential address
       // -- module instatiations --
                                                                                                                               // - wraps back to inst_base
20
                                                                                                                        20
21
                                                                                                                       21
22
                                                                                                                       22
                                                                                                                              wire tip eq end of mem = (tip q == 12'hfff);
23
                                                                                                                       23
                                                                                                                               wire tip_eq_pix_base = (tip_q == inst_base_pix);
24
                                                                                                                        24
       // -- combinational logic --
                                                                                                                               wire vtx_wrap = (cfs_thread_type == VTX) & tip_eq_pix_base;
                                            Page 7 of 20
                                                                                                                                                                    Page 8 of 20
                                                              Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                      Ex. 2102 - sq_target_instr_fetch.v
```

```
wire pix_wrap = (cfs_thread_type == PIX) & tip_eq_end_of_mem;
                                                                                                                                                    else
                                                                                                                                                                 tic q <= tic q;
 2
                                                                                                                                               end
 3
        always @(posedge clk)
 4
             if ( \ ld\_tip ) \qquad tip\_q \mathrel{<=} cfs\_instr\_ptr; \\
             else if ( inc_tip )
                                                                                                                                             // -- Input Staging Register --
              if ( vtx_wrap ) tip_q <= inst_base_vtx;
              else if ( pix_wrap ) tip_q <= inst_base_pix;
                                                                                                                                             // - register in control packet, vector type, and pc_base from the Arbiter
                         tip_q \le tip_q + 1;
                                                                                                                                             // when starting a new series of instruction fetches
                             tip\_q \mathrel{<=} tip\_q;
11
                                                                                                                                     11
                                                                                                                                             always @(posedge clk)
12
                                                                                                                                               begin
13
                                                                                                                                                    if (cfs_rts & tif_rtr)
14
                                                                                                                                                     begin
15
        // -- Target Instruction Counter (TIC) --
                                                                                                                                      15
                                                                                                                                                      isr export info q <= cfs export info;
16
                                                                                                                                      16
                                                                                                                                                     isr pc base q <= cfs pc base;
17
                                                                                                                                      17
                                                                                                                                                     isr_ctl_pkt_q <= cfs_ctl_pkt;
18
        // - loaded with reg'd in count at the same time TIP is loaded with the initial IS addr
                                                                                                                                      18
                                                                                                                                                    isr thread type q <= cfs thread type;
19
        // - dec'd by 1 for every instruction fetched
                                                                                                                                      19
                                                                                                                                                    isr thread id q <= cfs thread id;
                                                                                                                                                    isr last in thread q <= cfs last in thread;
20
                                                                                                                                      20
21
        always @(posedge clk)
                                                                                                                                     21
22
        begin
                                                                                                                                      22
23
              if (reset) \qquad tic\_q \mathrel{<=} 0; \\
                                                                                                                                     23
24
              else \ if (ld\_tip) \ \ tic\_q <= cfs\_instr\_cnt;
                                                                                                                                      24
                                                                                                                                                      isr_export_info_q <= isr_export_info_q;
              else \ if \ (dec\_tic) \ \ tic\_q \mathrel{<=} tic\_q - 1;
                                                                                                                                      25
                                                                                                                                                       isr\_pc\_base\_q <= isr\_pc\_base\_q;
                                                  Page 9 of 20
                                                                                                                                                                                       Page 10 of 20
                                                                     Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                                           Ex. 2102 - sq_target_instr_fetch.v
                isr_ctl_pkt_q <= isr_ctl_pkt_q;
                                                                                                                                                    tif_last_in_thread_q \le isr_last_in_thread_q;
 2
                                                                                                                                       2
              isr_thread_type_q <= isr_thread_type_q;
                                                                                                                                                    tif thread type q <= isr thread type q;
                 isr thread id q <= isr thread id q;
                                                                                                                                                    tif thread id q <= isr thread id q;
              isr\_last\_in\_thread\_q \mathrel{<=} isr\_last\_in\_thread\_q;
               end
         end
 7
                                                                                                                                                     tif\_export\_info\_q \  \  <= tif\_export\_info\_q;
                                                                                                                                                     tif_pc_base_q <= tif_pc_base_q;
                                                                                                                                                     tif\_ctl\_pkt\_q \le tif\_ctl\_pkt\_q;
10
        // -- Output Staging Register --
                                                                                                                                      10
                                                                                                                                                    tif\_last\_in\_group\_q \mathrel{<=} tif\_last\_in\_group\_q;
11
                                                                                                                                     11
                                                                                                                                                    tif\_last\_in\_thread\_q \mathrel{<=} tif\_last\_in\_thread\_q;
        /\!/ - hold misc info that must be passed on to the TIQ along with the instruction that was
                                                                                                                                     12
                                                                                                                                                    tif\_thread\_type\_q \mathrel{<=} tif\_thread\_type\_q;
                                                                                                                                     13
                                                                                                                                                      tif\_thread\_id\_q \mathrel{<=} tif\_thread\_id\_q;
       // - register ctl packet, type, cfs num, and last flag when loading the TIR
                                                                                                                                     14
        // - this is a stallable pipeline stage - stall when TIQ is not ready (input staging register can in
                                                                                                                                     15
17
        // mean time be loaded with new cfs data)
                                                                                                                                      17
18
                                                                                                                                             // -- Target Instruction Register (TIR) --
19
        always @(posedge clk)
20
         begin
                                                                                                                                     20
                                                                                                                                             // - loaded with data read from instruction store
21
              if (ld tir)
                                                                                                                                              /\!/ - the TIR is output to the target instruction queue (which does some decode in front of the
22
23
                tif\_export\_info\_q \  \  <= isr\_export\_info\_q;
                                                                                                                                     23
24
               tif\_pc\_base\_q \quad <= isr\_pc\_base\_q;
                                                                                                                                     24
                                                                                                                                              always @(posedge clk)
25
               tif\_ctl\_pkt\_q \  \, <= isr\_ctl\_pkt\_q;
                                                                                                                                     25
              tif\_last\_in\_group\_q \mathrel{<=} last\_in\_group;
                                                                                                                                                   if (ld\_tir) \ tif\_instr\_q <= is\_read\_data; \\
                                                 Page 11 of 20
                                                                                                                                                                                      Page 12 of 20
                                                                     Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                                           Ex. 2102 - sq target instr fetch.v
```

```
1
             else tif_instr_q <= tif_instr_q;
 2
                                                                                                                                2
         end
                                                                                                                                       always @(posedge clk)
                                                                                                                                           if (reset) \ tif\_instr\_rts\_q1 \mathrel{<=} LO; \\
                                                                                                                                           else tif_instr_rts_ql <= tif_instr_rts_q;
       // -- one-bit state machines --
       // - sets and clears the valid (rts) bit for the TIR
       /\!/ - the TIR can be valid and the TIQ not ready to take it when the next series of fetches is
13
       // (i.e. in Input Staging Reg can be reloaded while the Output Staging Reg still has the last
14
       // instruction of the previous cfs)
                                                                                                                                   /\!/ - load TIP with main_base + context_base + offset when starting a series of instruction fetches
15
16
       always @(posedge clk)
                                                                                                                               16
                                                                                                                                      // - read IS using TIP as read address
17
                                                                                                                                      // - load IS read data into TIR
                                                                                                                                      // - decrement TIC and increment TIP with every load until count equals zero
18
             if (reset) tif_instr_rts_q <= LO;
19
                                                                                                                               19
                                                                                                                                      parameter IDLE = 2'b00;
20
              case (tif_instr_rts_q)
                                                                                                                               20
21
               LO: tif\_instr\_rts\_q \mathrel{<=} ld\_tir; \qquad \textit{// - set when loading TIR}
                                                                                                                                      parameter LD_TIP = 2'b01;
22
               HI: tif\_instr\_rts\_q \mathrel{<=} \sim tiq\_rtr; \qquad \textit{// - clear when decoder/IQ is rdy}
                                                                                                                                      parameter FETCH = 2'b10;
23
                                                                                                                              23
                                                                                                                                      parameter LD_TIR = 2'b11;
24
                                                                                                                              24
25
       // reg delayed version for busy
                                              Page 13 of 20
                                                                                                                                                                             Page 14 of 20
                                                                 Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                                 Ex. 2102 - sq_target_instr_fetch.v
       reg next_ld_tir;
                                                                                                                                        end
                                                                                                                               2
 2
       reg next_last_in_group;
                                                                                                                                      // next state logic
                                                                                                                               3
 4
       // un-reg'd outputs
                                                                                                                                      //always @(*)
       //reg ld_tip;
                                                                                                                                      always @(
       //reg inc_tip;
                                                                                                                                           cfs_rts or alu_phase or is_phase or target_strap or tif_instr_rts_q or tic_q or
                                                                                                                                7
       //reg dec_tic;
                                                                                                                                              current_state
                                                                                                                                8
       // state and output registers
10
       always @(posedge clk)
                                                                                                                               10
                                                                                                                                          // default assignments
11
                                                                                                                              11
                                                                                                                                         next_state = IDLE;
12
                                                                                                                               12
                                                                                                                                          next_tif_rtr = LO;
13
                                                                                                                                           next_ld_tir = LO;
14
             current_state <= IDLE;
                                                                                                                                          next_last_in_group = LO;
15
             tif_rtr \le HI;
               ld_tir <= LO;
                                                                                                                                          ld_tip = LO;
17
             last_in_group <= LO;
                                                                                                                               17
                                                                                                                                          inc_tip = LO;
18
                                                                                                                                          dec_tic = LO;
19
20
                                                                                                                               20
                                                                                                                                           case (current state)
21
             current_state <= next_state;
                                                                                                                                            IDLE:
                                                                                                                              21
22
             tif rtr <= next tif rtr:
                                                                                                                               22
23
              ld tir <= next ld tir;
                                                                                                                               23
                                                                                                                                              // - send rdy back to the arbiter/ctl flow mgr while waiting for cfs rts
24
             last_in_group <= next_last_in_group;
                                                                                                                               24
                                                                                                                                              // - the inputs will be registered on cfs_rts
25
                                                                                                                                               // - kick off the state machine and deassert tif rdy on cfs_rts
                                              Page 15 of 20
                                                                                                                                                                             Page 16 of 20
                                                                 Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                                                Ex. 2102 - sq_target_instr_fetch.v
```

```
// 001
                                                                                                                                                    TEX - note that alu_phase is tied low on tex instance
 2
               next_tif_rtr = HI;
                                                                                                                                  // 010
                                                                                                                                                    ALU0
                                                                                                                                  // 110
                                                                                                                                                    ALUI
               if (cfs_rts)
                                                                                                                    4
                                                                                                                                  // - also make sure the TIR does not still contain valid data (if it does, then we don't
                ld_tip = HI;
                                                                                                                                  // overwrite it by loading new data from the IS)
                next_tif_rtr = LO;
              next_state = FETCH;
                                                                                                                                  if (\ (\{is\_phase, alu\_phase\} == target\_strap) \ \& \ (\sim tif\_instr\_rts\_q) \ )
                                                                                                                                  begin
10
                                                                                                                                    inc_tip = HI;
11
                                                                                                                                   dec_tic = HI;
12
                                                                                                                                  next_state = LD_TIR;
13
          begin
                                                                                                                   14
14
                                                                                                                   15
15
            // - wait one cycle for instr_ptr to be reg'd into this module
                                                                                                                   16
                                                                                                                                   begin
16
            //ld_{tip} = HI;
                                                                                                                                  next_state = FETCH;
                                                                                                                   17
17
            //next_state = FETCH;
                                                                                                                   18
                                                                                                                                 end
18
                                                                                                                   19
19
                                                                                                                   20
20
            FETCH:
                                                                                                                   21
                                                                                                                                LD TIR:
21
                                                                                                                   22
22
            // - synch with instr store phase (wait until phase == hardwired type)
                                                                                                                   23
                                                                                                                                  // - load TIR at the end of the next cycle
23
                                                                                                                   24
                                                                                                                                  // - if TIC is zero, we're done; otherwise, fetch another instruction
24
            // {is_phase, alu_phase} type
                                                                                                                   25
                                                                                                                                  next_ld_tir = HI;
                                          Page 17 of 20
                                                                                                                                                             Page 18 of 20
                                                            Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                               Ex. 2102 - sq_target_instr_fetch.v
 2
            if (tic_q == 0)
               begin
               next_last_in_group = HI;
              next_tif_rtr = HI;
               next_state = IDLE;
 7
              end
            else
              begin
              next_state = FETCH;
10
                                                                                                                   10
11
                                                                                                                   11
12
13
14
            endcase // case(current_state)
15
        end // always @ (*)
17
18
19
20
     endmodule
21
22
23
24
25
                                          Page 19 of 20
                                                                                                                                                             Page 20 of 20
                                                           Ex. 2102 - sq_target_instr_fetch.v
                                                                                                                                                                               Ex. 2102 - sq_target_instr_fetch.v
```

```
1 'include "header.v"
                                                                                                                      1 'include "sq_defs.v"
                                                                                                                      2
 2 //-----
                                                                                                                      3 module sq_export_alloc
 4 \hspace{0.5cm} /\hspace{0.1cm}/\hspace{0.1cm} \$ Id: /\hspace{-0.1cm}/\hspace{0.1cm} depot/r400/devel/parts\_lib/src/gfx/sq/misc/sq\_export\_alloc.v\#13~\$
                                                                                                                      4 (
 5 //
                                                                                                                      5 // - inputs from local registers
                                                                                                                          \label{eq:vs_export_count_set} vs\_export\_count\_set, \qquad \mbox{$/$} // \mbox{ connected to $SQ\_PROGRAM\_CNTL.VS\_EXPORT\_COUNT} \mbox{$(4$ bits)$}
 6 // $Change: 44314 $
 7 //
                                                                                                                           \label{local_vs_prop} $$ vs_export_mode_set, \qquad //\ connected \ to \ SQ_PROGRAM_CNTL.VS_EXPORT_MODE \ (3 \ bits) $$
 8 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                                           \label{eq:program_cnt_mode_set} ps\_export\_mode\_set, \qquad \mbox{$//$ connected to SQ_PROGRAM\_CNTL.PS\_EXPORT\_MODE (3 bits)}
                  © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
10 //
                                                                                                                     12
11 //
                All rights reserved. This notice is intended as a precaution against
                                                                                                                     13
                                                                                                                            alu_arb_rts0,// ready to send the winner to CFS0
                inadvertent publication and does not imply publication or any waiver
                                                                                                                     14 alu arb rts1,// ready to send the winner to CFS1
13 //
                of confidentiality. The year included in the foregoing notice is the
                                                                                                                     15 alu_arb_context_id, // the state sent to the CFS
14 //
                year of creation of the work.
                                                                                                                     16 alu_arb_status, // the status sent to the CFS
15 //
                                                                                                                     17 alu arb thread type, // vtx or pix
                                                                                                                     18
19
                                                                                                                            alu0_cfs_rtr,// ALU_CFS0 can accept a thread
18 // sq export alloc.v
                                                                                                                     20
                                                                                                                           alu1_cfs_rtr,// ALU_CFS1 can accept a thread (for alu cfs's)
19 //
                                                                                                                     21
20 // -
                                                                                                                     22 pb_dealloc_cnt, // param cache dealloc info from SC via the sq_ptr_buff
21 //
                                                                                                                     23 pb_dealloc_vld,
                                                                                                                     24 param_cache_wptr_q, // output to SX and status regs
23 // -
24 //
                                                                                                                           // - sx export alloc interface
Page 1 of 18
                                                                                                                                                                Page 2 of 18
                                                                 Ex. 2103 - sq export alloc.y
                                                                                                                                                                                       Ex. 2103 - sq export alloc.y

    SQ_SX_exp_valid,

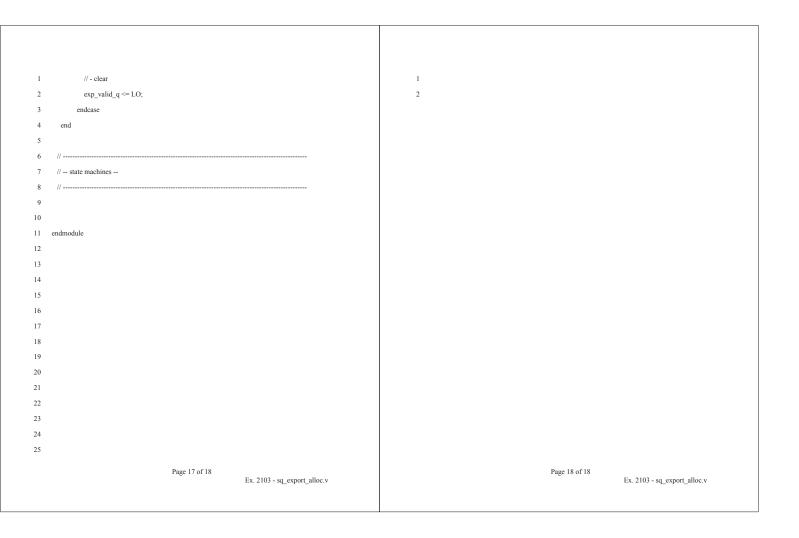
                                                                                                                           reset
                                                                                                                      2 );
      SQ_SX_exp_type,
       SQ_SX_exp_number,
      SQ_SX_exp_context_id,
 5 SQ_SX_exp_id,
                                                                                                                      6 parameter LO = 1'b0;
 7 ais0_free_done,
                                                                                                                      7 parameter HI = 1'b1;
 8 ais0 free id,
                                                                                                                            parameter X = 1'bx;
       ais1 free done,
11
                                                                                                                     11
12 // - sx export dealloc interface
                                                                                                                     12
13 SQ_SX_free_done,
14 SQ_SX_free_id,
                                                                                                                     14
                                                                                                                     15
15
16
                                                                                                                            input [8*4-1:0] vs_export_count_set;
                                // export_id that cfs0 is pushing down pipe 0 (sets global
                                                                                                                     17
                                                                                                                           input [8*3-1:0] vs_export_mode_set;
                                                                                                                     18 input [8*4-1:0] ps_export_mode_set;
19 cfs aif xfc0,// cfs0 to aif0 transfer complete
20 _ cfs1_export_id, ____// export_id that cfs1 is pushing down pipe 1 (sets global 21 _ export_id)
                                                                                                                     20 input [0:0]
                                                                                                                                                     alu_arb_rts0;
                                                                                                                     21 input [0:0]
                                                                                                                                                    alu arb rts1:
22 \qquad cfs\_aif\_xfc1,\!/\!/\,cfs1 \ to \ aif1 \ transfer \ complete
                                                                                                                                                     alu_arb_context_id;
23
                                                                                                                            input [`SQ_STATUS_WIDTH-1:0] alu_arb_status;
      global_export_id_q,// exp_id taken on arb_xfc, toggled on exp_valid, loaded on cfs_xfc
25
                                                                                                                     24
                                                                                                                            input [0:0]
                                                                                                                                                    alu_arb_thread_type;
                                                                                                                            input [0:0]
                                                                                                                                                     alu0_cfs_rtr;
26 clk.
                                           Page 3 of 18
                                                                                                                                                               Page 4 of 18
                                                                 Ex. 2103 - sq export alloc.v
                                                                                                                                                                                      Ex. 2103 - sq_export_alloc.v
```

ATI 2103 LG v. ATI IPR2015-00326

```
input [0:0]
                                alul efs rtr
                                                                                                                            input [0:0] cfs aif xfc1;
 2
 3
       input [2:0] pb dealloc ent;
                                                                                                                           input [0:0] pb_dealloc_vld;
       output [6:0] param_cache_wptr_q;
                                                                                                                            input clk;
                                                                                                                            input reset;
       //output [0:0] SQ_SX_exp_pix;
       output [0:0] SQ_SX_exp_valid;
       output [1:0] SQ_SX_exp_type;
                                                                                                                     11
                                                                                                                            // -- internal signals --
11
       output [1:0] SQ_SX_exp_number;
                                                                                                                     12
12
       output [2:0] SQ_SX_exp_context_id;
                                                                                                                     13
13
       output [0:0] SQ SX exp id;
                                                                                                                     14
                                                                                                                            reg [0:0] exp valid q; // one bit state machine
14
                                                                                                                     15
15
       input [0:0] ais0 free done;
                                                                                                                     16
16
       input [0:0] ais0 free id:
                                                                                                                     17
17
       input [0:0] ais1 free done;
                                                                                                                     18
                                                                                                                            // -- module instatiations --
18
       input [0:0] ais1_free_id;
                                                                                                                     19
19
                                                                                                                     20
20
       output [0:0] SQ SX free done;
                                                                                                                     21
                                                                                                                            parameter VS_EXPORT_COUNT_WIDTH = 4;
21
       output [0:0] SQ_SX_free_id;
                                                                                                                     22
                                                                                                                            wire [VS_EXPORT_COUNT_WIDTH-1:0] vs_export_count;
22
                                                                                                                     23
                                                                                                                            reg [VS_EXPORT_COUNT_WIDTH-1:0] vs_export_count_q;
23
       input [0:0] cfs0_export_id;
                                                                                                                     24
                                                                                                                            sq_state_mux #( VS_EXPORT_COUNT_WIDTH )
24
       input [0:0] cfs_aif_xfc0;
                                                                                                                           vs_export_count_sel (.state(alu_arb_context_id), .mux_data_out(vs_export_count));
                                                                                                                                                                                   .input_set(vs_export_count_set),
25
       input [0:0] cfs1_export_id;
                                           Page 5 of 18
                                                                                                                                                                 Page 6 of 18
                                                                 Ex. 2103 - sq_export_alloc.v
                                                                                                                                                                                       Ex. 2103 - sq_export_alloc.v
       parameter VS_EXPORT_MODE_WIDTH = 3;
 2
                                                                                                                      2
                                                                                                                            wire [0:0] alu arb xfc0 = alu arb rts0 & alu0 cfs rtr;
       wire [VS_EXPORT_MODE_WIDTH-1:0] vs_export_mode;
                                                                                                                            wire [0:0] alu arb xfc1 = alu arb rts1 & alu1 cfs rtr;
       reg [VS_EXPORT_MODE_WIDTH-1:0] vs_export_mode_q;
                                                                                                                            wire [0:0] alu_arb_xfc = alu_arb_xfc0 | alu_arb_xfc1;
       sq_state_mux #( VS_EXPORT_MODE_WIDTH )
      vs_export_mode_sel (.state
.mux_data_out(vs_export_mode));
                             (.state(alu_arb_context_id),
                                                              .input_set(vs_export_mode_set),
                                                                                                                            // - alloc type/size status to SX export type/number decode
                                                                                                                            wire [0:0] thread_type = alu_arb_thread_type;
       parameter PS_EXPORT_MODE_WIDTH = 4;
                                                                                                                            wire [1:0] alloc_type = alu_arb_status[9:8];
10
       wire [PS_EXPORT_MODE_WIDTH-1:0] ps_export_mode;
                                                                                                                     10
                                                                                                                            wire [3:0] alloc_size = alu_arb_status[7:4];
11
       reg [PS_EXPORT_MODE_WIDTH-1:0] ps_export_mode_q;
                                                                                                                     11
12
        sq_state_mux #( PS_EXPORT_MODE_WIDTH )
                                                                                                                     12
                                                                                                                            wire~[7:0]~alloc\_cmd = \{thread\_type,~alloc\_type,~alloc\_size,~ps\_export\_mode\_q[0]\};
      ps_export_mode_sel (.state(alu_arb_context_id), .mux_data_out(ps_export_mode));
                                                            .input set(ps export mode set),
                                                                                                                     13
                                                                                                                     14
15
                                                                                                                            reg [4:0] sx_exp_cmd;
                                                                                                                     15
16
       always @(posedge clk)
                                                                                                                            always @( alloc_cmd )
17
        begin
                                                                                                                     17
                                                                                                                             begin
18
            vs_export_count_q <= vs_export_count;
                                                                                                                     18
                                                                                                                                 casez ( alloc_cmd )
19
            vs_export_mode_q <= vs_export_mode;
                                                                                                                                  // - vtx pos alloc
20
            ps\_export\_mode\_q \mathrel{<=} ps\_export\_mode;
                                                                                                                                  8'b1 01 0000 ?: sx exp cmd = 5'b10 00 1;
21
        end
                                                                                                                     20
                                                                                                                     21
                                                                                                                                  8'b1 01 0001 ?: sx exp cmd = 5'b10 01 1;
22
                                                                                                                     22
23
                                                                                                                     23
                                                                                                                                  // - vtx pass thru
24
                                                                                                                     24
                                                                                                                                  8'b1_11_0011_?: sx_exp_cmd = 5'b11_00_1;
25
        // -- combinational logic --
                                                                                                                     25
                                                                                                                                   8'b1\_11\_0111\_?: sx\_exp\_cmd = 5'b11\_01\_1;\\
                                           Page 7 of 18
                                                                                                                                                                 Page 8 of 18
                                                                 Ex. 2103 - sq_export_alloc.v
                                                                                                                                                                                       Ex. 2103 - sq_export_alloc.v
```

```
8'b1 11 1011 ?: sx exp cmd = 5'b11 10 1;
                                                                                                                                       wire [1:0] exp_type = sx_exp_cmd[4:3];
 2
                                                                                                                                       wire [1:0] exp_number = sx_exp_cmd[2:1];
              // - pix without z
                                                                                                                                       wire [0:0] exp_valid = sx_exp_cmd[0];
              8'b0_10_0000_0 : sx_exp_cmd = 5'b00_00_1;
              8'b0\_10\_0001\_0: sx\_exp\_cmd = 5'b00\_01\_1; \\
                                                                                                                                      // also need to decode for param cache alloc
              8'b0\_10\_0010\_0: sx\_exp\_cmd = 5'b00\_10\_1; \\
              8'b0_10_0011_0 : sx_exp_cmd = 5'b00_11_1;
                                                                                                                                       wire [0:0] pc_alloc = (thread_type == HI) & (alloc_type == 2'b10) & alu_arb_xfc;
10
              8'b0_10_0001_1 : sx_exp_cmd = 5'b01_00_1;
                                                                                                                                       // -- registers --
11
              8'b0_10_0010_1 : sx_exp_cmd = 5'b01_01_1;
12
              8'b0_10_0011_1 : sx_exp_cmd = 5'b01_10_1;
                                                                                                                               12
13
              8'b0_10_0100_1 : sx_exp_cmd = 5'b01_11_1;
                                                                                                                               13
                                                                                                                                      // -- global export id --
14
                                                                                                                                      // - toggled when ID is assigned to a thread (same as when alloc is sent to SX)
15
                                                                                                                                      // - overwritten when thread leaves the CFS and enters the ALU Instr Fetch
              // - pix pass thru
                                                                                                                               15
              8'b0_11_0011_?: sx_exp_cmd = 5'b11_00_1;
16
                                                                                                                               16
17
              8'b0_11_0111_?: sx_exp_cmd = 5'b11_01_1;
                                                                                                                               17
                                                                                                                                       reg [0:0] global export id q;
              8'b0_11_1011_?: sx_exp_cmd = 5'b11_10_1;
18
                                                                                                                               18
19
                                                                                                                               19
                                                                                                                                       always @(posedge clk)
20
              default: sx exp cmd = 5'bxxxx0;
                                                                                                                               20
21
             endcase
                                                                                                                               21
                                                                                                                                            if (reset)
                                                                                                                                                              global\_export\_id\_q \mathrel{<=} LO;
22
         end
                                                                                                                               22
                                                                                                                                             else\ if\ (\ cfs\_aif\_xfc0\ )\ \ global\_export\_id\_q <= cfs0\_export\_id;
23
                                                                                                                               23
                                                                                                                                            else\ if\ (\ cfs\_aif\_xfc1\ )\ \ global\_export\_id\_q <= cfs1\_export\_id;
24
       //wire [0:0] exp_pix = sx_exp_cmd[5];
                                                                                                                               24
                                                                                                                                            else \ if \ (\ exp\_valid\_q \ ) \quad global\_export\_id\_q <= \neg global\_export\_id\_q;
25
                                                                                                                               25
                                                                                                                                                             global\_export\_id\_q \mathrel{<=} global\_export\_id\_q;
                                               Page 9 of 18
                                                                                                                                                                              Page 10 of 18
                                                                       Ex. 2103 - sq_export_alloc.v
                                                                                                                                                                                                       Ex. 2103 - sq_export_alloc.v
         end
2
                                                                                                                                2
                                                                                                                                       always @(posedge clk)
                                                                                                                                        begin
4
       // - reg delay arb xfc for load of SQ SX export alloc interface output registers
                                                                                                                                             if (reset)
                                                                                                                                                              param_cache_wptr_q <= 7'b0;
                                                                                                                                     else if ( pc_alloc ) vs_export_count_x4;
5
                                                                                                                                                                        param_cache_wptr_q <= param_cache_wptr_q +
       reg [0:0] alu arb xfc q;
                                                                                                                                      else \ \ if \ ( \ pb\_dealloc\_vld \ \ ) \ \ param\_cache\_wptr\_q \ <= \ param\_cache\_wptr\_q \ - pb\_dealloc\_cnt\_x4;
        always @(posedge clk)
                                                                                                                                                              param\_cache\_wptr\_q \mathrel{<=} param\_cache\_wptr\_q;
            alu_arb_xfc_q <= alu_arb_xfc;
10
                                                                                                                               11
11
12
13
                                                                                                                                       // -- SQ_SX_exp output registers --
14
                                                                                                                               15
        // -- param cache write pointer --
                                                                                                                               16
16
                                                                                                                               17
                                                                                                                                      //reg [0:0] SQ_SX_exp_pix;
17
                                                                                                                               18
        // - multiply the counts by 4 (left shift 2x) since there are 16 verts per P$ line, and thus 4 lines
                                                                                                                               19
                                                                                                                                       reg [0:0] SQ_SX_exp_valid;
19
                                                                                                                               20
                                                                                                                                       reg [1:0] SQ_SX_exp_type
       \ensuremath{/\!/} - the vs_export count ranges from 0 to 15 for 1 to 16 params, so need to add 1 to get the true
20
21
                                                                                                                               21
                                                                                                                                       reg~[1:0]~SQ\_SX\_exp\_number;
22
                                                                                                                               22
                                                                                                                                       reg~[0:0]~SQ\_SX\_exp\_id;
23
       reg [6:0] param_cache_wptr_q;
                                                                                                                               23
                                                                                                                                       reg [2:0] SQ_SX_exp_context_id;
24
                                                                                                                               24
25
        wire [6:0] vs_export_count_x4 = (vs_export_count_q + 1) << 2; // 4 + 1 + 2 = 7 bits
                                                                                                                               25
                                                                                                                                       always @(posedge clk)
        wire [4:0] pb_dealloc_cnt_x4 = pb_dealloc_cnt \ll 2;  // 3 + 2 = 5 bits
                                               Page 11 of 18
                                                                                                                                                                              Page 12 of 18
                                                                       Ex. 2103 - sq_export_alloc.v
                                                                                                                                                                                                       Ex. 2103 - sq_export_alloc.v
```

```
2
           SQ\_SX\_exp\_valid <= exp\_valid\_q;
                                                                                                                                    reg [0:0] free_done_q0;
                                                                                                                                    reg [0:0] free_id_q0;
             if ( alu_arb_xfc )
                                                                                                                                    reg [0:0] free_done_q1;
             /\!/ SQ\_SX\_exp\_pix <= \sim\!exp\_pix;
                                                                                                                                    reg~[0:0]~free\_id\_q1;
             SQ_SX_exp_context_id <= alu_arb_context_id;
             SQ_SX_exp_id <= alu_arb_xfc1;
                                                                                                                                    reg [0:0] free_done_q2;
                                                                                                                                    reg [0:0] free_id_q2;
             SQ_SX_exp_type <= exp_type;
             SQ\_SX\_exp\_number <= exp\_number;
11
                                                                                                                                   reg [0:0] free_done_q3;
12
                                                                                                                                    reg [0:0] free_id_q3;
13
                                                                                                                            13
14
             //SQ_SX_exp_pix <= SQ_SX_exp_pix;
                                                                                                                                    reg [0:0] free_done_q4;
15
                                                                                                                            15
                                                                                                                                    reg [0:0] free_id_q4;
             SQ SX exp context id <= SQ SX exp context id;
                                                                                                                            16
16
             SQ SX exp id <= SQ SX exp id;
17
             SQ_SX_exp_type <= SQ_SX_exp_type;
                                                                                                                            17
                                                                                                                                    reg [0:0] free_done_q5;
18
             SQ\_SX\_exp\_number <= SQ\_SX\_exp\_number;
                                                                                                                            18
                                                                                                                                    reg [0:0] free_id_q5;
19
                                                                                                                            19
20
                                                                                                                            20
                                                                                                                                    reg [0:0] free_done_q6;
                                                                                                                            21
21
                                                                                                                                    reg [0:0] free_id_q6;
22
       // - delay free done and id to line up with last cycle of data out of the SP
                                                                                                                            22
23
                                                                                                                            23
                                                                                                                                    reg [0:0] free_done_q7;
24
       reg [0:0] SQ_SX_free_done;
                                                                                                                            24
                                                                                                                                    reg [0:0] free_id_q7;
25
       reg [0:0] SQ_SX_free_id;
                                                                                                                            25
                                             Page 13 of 18
                                                                                                                                                                          Page 14 of 18
                                                                     Ex. 2103 - sq_export_alloc.v
                                                                                                                                                                                                  Ex. 2103 - sq_export_alloc.v
       reg [0:0] free_done_q8;
                                                                                                                                       free_id_q5 <= free_id_q4;
       reg [0:0] free_id_q8;
 2
                                                                                                                                       free id q6 <= free id q5;
                                                                                                                                       free_id_q7 <= free_id_q6;
 4
       reg [0:0] free_done_q9;
                                                                                                                                       free_id_q8 \le free_id_q7;
                                                                                                                                       free\_id\_q9 \mathrel{<=} free\_id\_q8;
       reg [0:0] free_id_q9;
                                                                                                                                       SQ_SX_free_id \le free_id_q9;
 7
       always @(posedge clk)
 8
           free\_done\_q0 \mathrel{<=} ais0\_free\_done \mid ais1\_free\_done;
10
           free\_done\_q1 <= free\_done\_q0;
                                                                                                                            10
11
           free\_done\_q2 \mathrel{<=} free\_done\_q1;
                                                                                                                            11
                                                                                                                                    // -- one-bit state machines --
12
           free\_done\_q3 \mathrel{<=} free\_done\_q2;
                                                                                                                            12
13
           free\_done\_q4 \le free\_done\_q3;
                                                                                                                            13
14
           free_done_q5 <= free_done_q4;
                                                                                                                                    // exp_valid
15
           free_done_q6 <= free_done_q5;
                                                                                                                            15
16
           free\_done\_q7 \mathrel{<=} free\_done\_q6;
                                                                                                                                    always @(posedge clk)
17
           free\_done\_q8 \mathrel{<=} free\_done\_q7;
                                                                                                                            17
                                                                                                                                     begin
18
           free_done_q9 <= free_done_q8;
                                                                                                                            18
                                                                                                                                         if (reset)
19
           SQ_SX_free_done <= free_done_q9;
                                                                                                                                          exp_valid_q <= LO;
20
                                                                                                                            20
21
           free_id_q0 <= ais1_free_done ? ais1_free_id : ais0_free_id;
                                                                                                                            21
                                                                                                                                         case (exp valid q)
22
           free_id_q1 \le free_id_q0;
                                                                                                                            22
                                                                                                                                        LO:
23
           free_id_q2 \le free_id_q1;
                                                                                                                            23
                                                                                                                                           // - set
24
           free\_id\_q3 \mathrel{<=} free\_id\_q2;
                                                                                                                            24
                                                                                                                                            exp\_valid\_q \mathrel{<=} exp\_valid \& alu\_arb\_xfc;
           free\_id\_q4 \mathrel{<=} free\_id\_q3;
                                                                                                                            25
                                             Page 15 of 18
                                                                                                                                                                          Page 16 of 18
                                                                                                                                                                                                  Ex. 2103 - sq_export_alloc.v
                                                                     Ex. 2103 - sq_export_alloc.v
```



1 // -*- Mode: Verilog -	*-	1);		
· ·	1-		,	
2 // Filename : vector.v		2		
3 // Description : This module represen	t the implementation of the vector unit.	3 //	//	
4 // There are 4 macc_reg mod	dules instantiated from this module.	4 //	/Instruction Interface	
5 // Author : Andi Skende		5 //	//	
6 // Created On : Wed Jan 30 15:51:38	2002	6 is	nnut [0:0] ca an instruct start	
	2002		nput [0:0] sq_sp_instruct_start;	
7 // Last Modified By: .		7 ir	nput [20:0] sq_sp_instruct; //four cycle transaction	
8 // Last Modified On: .		8 ir	nput [0:0] sq_sp_stall; //when highexecute a NOP	
9 // Update Count : 0		9 ir	nput sclk;	
10 // Status : Unknown, Use with cau	tion!	10 ir	nput srst;	
11			1	
		11		
12 // timescale 1ns / 1ps		12 //	///	/
13		13 //	Export controls signals	
14 module vector(/*AUTOARG*/		14 //	/	//
15 // Outputs		15 ir	nput [3:0] sq_sp_exp_pvalid;	
·				
sp_sx_data, sp_tp_data, sp_sx_exp_ds	si, sp_sx_exp_pvalid,		nput [0:0] sq_sp_exporting;	
17 sp_sx_exporting, sp_sx_exp_alu_id,		17 ir	nput [0:0] sq_sp_exp_alu_id;	
18 // Inputs		18		
19 sq_sp_instruct_start, sq_sp_instruct, s	q_sp_stall, sclk, srst,	19 //	export destination pointer/	
20 sq_sp_exp_pvalid, sq_sp_exporting, s			vire [5:0] sq_sp_exp_dst;	
212 12 1 212 1 0			[] od_op_oop_oos,	
21 sq_sp_wr_addr, sq_sp_gpr_rd_addr, s	sq_sp_gpr_phase_mux,	21		
22 sq_sp_channel_mask, sq_sp_pixel_ma	ask, sq_sp_gpr_input_mux,	22 //	//	
23 sq_sp_mem_rd_ena, sq_sp_mem_wr_	_ena, sq_sp_wr_ena, iInterpolated,	23 //	/GPR read/write control interface	
24 iAutoCount, sq_sp_constant, tp_sp_da	ata, tp sp gpr dst,	24 //	//	
25 tp sp gpr cmask, tp sp data valid, i		25 ir	nput [6:0] sq_sp_wr_addr;	
25 tp_sp_gpr_cmask, tp_sp_data_vand, r	vertexindices	2.5 11	iput [0.0] sq_sp_wr_addr,	
	Page 1 of 22		Page 2 of 22	
	Ex. 2104 - vector.v			Ex. 2104 - vector.v
<pre>1 input [6:0] sq_sp_gpr_rd_addr;</pre>		1		
 input [6:0] sq_sp_gpr_rd_addr; input [1:0] sq_sp_gpr_phase_mux; 			/	-/
2 input [1:0] sq_sp_gpr_phase_mux;		2 //		
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask;		2 //	Output data going out from each vector unit to the SX bloc	ks
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask; 4 input [3:0] sq_sp_pixel_mask;		2 // 3 // 4 //	Output data going out from each vector unit to the SX block.	ks
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux;		2 // 3 // 4 // 5 o	Output data going out from each vector unit to the SX block	ks
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask; 4 input [3:0] sq_sp_pixel_mask;		2 // 3 // 4 // 5 o	Output data going out from each vector unit to the SX block.	ks
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux;		2 // 3 // 4 // 5 o 6 re	Output data going out from each vector unit to the SX block	
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena;		2 // 3 // 4 // 5 o 6 re	Output data going out from each vector unit to the SX block	
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_channel_mask; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena;		2 // 3 // 4 // 5 o 6 re 7 w	Output data going out from each vector unit to the SX block	/ectorResult3;
2 input [1:0] sq.sp.ger_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_pixel_mask; 5 input [1:0] sq.sp_ger_input_mux; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9		2 // 3 // 4 // 5 o 6 rr 7 w 8	Output data going out from each vector unit to the SX bloc unutput [127-0] sp_sx_data; eg [127-0] osp_sx_data; vire [127-0] VectorResult0,VectorResult1,VectorResult2,V	/ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_ptr_phase_mux; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9		2 // 3 // 4 // 5 o 6 rr 7 w 8	Output data going out from each vector unit to the SX bloc untput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult1,VectorResult2,V under [127:0] VectorResult0,VectorResult1,VectorResult2,V under [127:0] VectorResult0,VectorResult0,VectorResult2,V under [127:0] VectorResult0,VectorResult0,VectorResult2,V under [127:0] VectorResult0,VectorResult0,VectorResult2,V under [127:0] VectorResult0,Vect	cks -/ -/ectorResult3;
2 input [1:0] sq.sp.ger_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_pixel_mask; 5 input [1:0] sq.sp_ger_input_mux; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9		2 // 3 // 4 // 5 o 6 r 7 w 8	Output data going out from each vector unit to the SX bloc unutput [127-0] sp_sx_data; eg [127-0] osp_sx_data; vire [127-0] VectorResult0,VectorResult1,VectorResult2,V	.cks / //ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_ptr_phase_mux; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9		2 // 3 // 4 // 5 o 6 rr 7 w 8 9 // 10 // 11 // 11 // 11	Output data going out from each vector unit to the SX bloc untput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult1,VectorResult2,V under [127:0] VectorResult0,VectorResult1,VectorResult2,V under [127:0] VectorResult0,VectorResult0,VectorResult2,V under [127:0] VectorResult0,VectorResult0,VectorResult2,V under [127:0] VectorResult0,VectorResult0,VectorResult2,V under [127:0] VectorResult0,Vect	cks -/ -/ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_epr_phase_mux; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9 10 //		2 // 3 // 4 // 5 o 6 \(\pi \) 7 \(w \) 8 \(\text{9} \) // 11 // 12 o	Output data going out from each vector unit to the SX bloc unitput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,V unitput [127:0] vectorResult0,VectorResult1,VectorResult2,V unitput data going out to the texture pipe	cks -/ -/ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_gpr_phase_mux; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 10 //		2 // 3 // 4 // 5 o 6 \(\pi \) 7 \(w \) 8 \(\text{9} \) // 11 // 12 o	Output data going out from each vector unit to the SX block unitput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,Ve	.cks / //ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_gpr_phase_mux; 4 input [3:0] sq_sp_pixel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9 10 //		2 // 3 // 4 // 5 o o 6 rr 7 w 8 9 // 11 // 12 o 13 rr 14	Output data going out from each vector unit to the SX block output [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult3, V	.cks / //ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_gpr_phase_mux; 4 input [3:0] sq_sp_ptr_lmask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_gpr_input_mux; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9	/	2 // 3 // 4 // 5 o o 6 rr 7 w 8 9 // 11 // 12 o 13 r 14	Output data going out from each vector unit to the SX block without [127:0] sp_sx_data; gg [127:0] osp_sx_data; vire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult3, VectorResult3	/cctorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_gpr_phase_mux; 4 input [3:0] sq_sp_ptr_lanask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_gpr_input_mux; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9	nt.sq_sp_constant;	2 // 3 // 4 // 5 o o 6 rr 7 w 8 9 // 11 // 12 o 13 rr 14 15 // 16 // 7	Output data going out from each vector unit to the SX block output [127:0] sp_sx_data; gg [127:0] osp_sx_data; vire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult4, VectorResult5, VectorResult6, VectorResult7, V	/ectorResult3;
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_gpr_phase_mux; 4 input [3:0] sq_sp_ptannel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_gpr_input_mux; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine	2 // 3 // 4 // 5 o 6 r 7 w 8 9 // 11 // 12 o 13 r 14 15 // 16 // 17 //	Output data going out from each vector unit to the SX block without [127:0] sp_sx_data; gg [127:0] osp_sx_data; wire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult4, VectorResult5, VectorResult6, VectorResult7,	kss -/ /ectorResult3;////
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_gpr_phase_mux; 4 input [3:0] sq_sp_ptannel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_gpr_input_mux; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 //	nt.sq_sp_constant;	2 // 3 // 4 // 5 o 6 r 7 w 8 9 // 11 // 12 o 13 r 14 15 // 16 // 17 //	Output data going out from each vector unit to the SX block output [127:0] sp_sx_data; gg [127:0] osp_sx_data; vire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult4, VectorResult5, VectorResult6, VectorResult7, V	kss -/ /ectorResult3;////
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_ptnnel_mask; 4 input [3:0] sq_sp_phanel_mask; 5 input [1:0] sq_sp_ptr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine	2 // 3 // 4 // 5 o 6 r 7 w 8 9 // 11 // 12 o 13 r 14 15 // 16 // 17 //	Output data going out from each vector unit to the SX block without [127:0] sp_sx_data; gg [127:0] osp_sx_data; wire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult4, VectorResult5, VectorResult6, VectorResult7,	kss -/ /ectorResult3;////
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_per_phase_mux; 4 input [3:0] sq_sp_pt_nanel_mask; 5 input [1:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_gpr_input_mux; 6 input [0:0] sq_sp_mem_wr_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_wr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data	2 // 3 // 4 // 5 o o 6 rr 7 w 8 9 // 11 // 12 o o 13 rr 14 15 // 16 // 17 // 18 //	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; gg [127:0] osp_sx_data; vire [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult3, VectorResult4, VectorResult4, VectorResult5, VectorResult6, VectorResult7, V	kss -/ /ectorResult3;////
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_ptnnel_mask; 4 input [3:0] sq_sp_phanel_mask; 5 input [1:0] sq_sp_ptr_input_mux; 6 input [0:0] sq_sp_ptr_input_mux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_yr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data	2 // 3 // 4 // 5 0 6 rr 7 w 8 9 // 10 // 11 // 12 0 13 rr 14 15 // 16 // 17 // 18 // 19 20 0	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; gg [127:0] osp_sx_data; gg [127:0] VectorResult0,VectorResult1,VectorResult2,V Register (GPR) data going out to the texture pipe witput [95:0] sp_tp_data; gg [95:0] osp_tp_data; Pipelined sequencer inputsrelated to export functionality without [55:0] sp_sx_exp_dst;	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_ptr_phase_mux; 4 input [3:0] sq_sp_ptr_lmak; 5 input [1:0] sq_sp_ptr_lmak; 6 input [0:0] sq_sp_ptr_lmux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data	2 // 3 // 4 // 4 // 5 o o 6 rs 5 o o 6 rs 5 o o 6 rs 6 rs 7 w 8 s 9 // 11 // 12 o o 13 rs 14 s 15 // 18 // 17 // 18 // 19 20 o o 21 o o o 6 o o f o o f o o f o o o o o o o	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,VectorResult2,VectorResult2,VectorResult3,VectorResult3,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,Vec	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_pixel_mask; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data unel mask for the texture data being written into gpr(s)	2 // 3 // 3 // 4 // 5 o o 6 rs 6 o o o o o o o o o o o o o o o o o o	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,VectorResult2,VectorResult2,VectorResult3,VectorResult3,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,VectorResult2,VectorResult4,VectorResult2,VectorResult4,Vec	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq_sp_gpr_phase_mux; 3 input [3:0] sq_sp_ptr_phase_mux; 4 input [3:0] sq_sp_ptr_lmak; 5 input [1:0] sq_sp_ptr_lmak; 6 input [0:0] sq_sp_ptr_lmux; 6 input [0:0] sq_sp_mem_rd_ena; 7 input [0:0] sq_sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data unel mask for the texture data being written into gpr(s)	2 // 3 // 3 // 4 // 5 o o 6 rs 5 // 3 // 3 // 3 // 3 // 3 // 3 // 3 /	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,VectorResult2,VectorResult2,VectorResult3,VectorResult3,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,Vec	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_pixel_mask; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq_sp_mem_wr_ena; 9 10 //	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data unel mask for the texture data being written into gpr(s)	2 // 3 // 3 // 4 // 5 o o 6 rs 6 o o o o o o o o o o o o o o o o o o	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,VectorResult2,VectorResult2,VectorResult3,VectorResult3,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,VectorResult2,VectorResult4,VectorResult2,VectorResult4,Vec	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_pixel_mask; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq.sp_mem_wr_ena; 9	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data unel mask for the texture data being written into gpr(s) pr_dst, q2_tp_gpr_dst;	2 // 3 // 3 // 4 // 5 0 0 6 rs 7 w 8 9 // 10 // 11 // 12 0 13 rs 14 15 // 16 // 17 // 18 // 19 20 0 21 0 22 0 23 0 24	Output data going out from each vector unit to the SX block witput [127:0] sp_sx_data; eg [127:0] osp_sx_data; vire [127:0] VectorResult0,VectorResult1,VectorResult2,VectorResult2,VectorResult2,VectorResult3,VectorResult3,VectorResult4,VectorResult2,VectorResult4,VectorResult4,VectorResult2,VectorResult4,VectorResult2,VectorResult4,VectorResult2,VectorResult4,Vec	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_pixel_mask; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq.sp_mem_wr_ena; 9	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data nnel mask for the texture data being written into gpr(s) pr_dst, q2_tp_gpr_dst; p_gpr_cmask, q2_tp_gpr_cmask; p_data_valid, q2_tp_data_valid;	2 // 3 // 3 // 4 // 5 0 0 6 rs 7 w 8 9 // 10 // 11 // 12 0 13 rs 14 15 // 16 // 17 // 18 // 19 20 0 21 0 22 0 23 0 24	Output data going out from each vector unit to the SX block content [127:0] sp_sx_data; gg [127:0] osp_sx_data; gg [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult2, VectorResult4, VectorResult4, VectorResult2, VectorResult4, VectorResult4, VectorResult2, VectorResult4, V	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_pixel_mask; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq.sp_mem_wr_ena; 9	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data unel mask for the texture data being written into gpr(s) pr_dst, q2_tp_gpr_dst; p_gpr_cmask, q2_tp_gpr_cmask; p_data_valid, q2_tp_data_valid; Page 3 of 22	2 // 3 // 3 // 4 // 5 0 0 6 rs 7 w 8 9 // 10 // 11 // 12 0 13 rs 14 15 // 16 // 17 // 18 // 19 20 0 21 0 22 0 23 0 24	Output data going out from each vector unit to the SX block John March (127:0) sp_sx_data; eg [127:0] osp_sx_data; eg [127:0] VectorResult(),VectorResult	// vectorResult3;///////////
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_pixel_mask; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq.sp_mem_wr_ena; 9	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data nnel mask for the texture data being written into gpr(s) pr_dst, q2_tp_gpr_dst; p_gpr_cmask, q2_tp_gpr_cmask; p_data_valid, q2_tp_data_valid;	2 // 3 // 3 // 4 // 5 0 0 6 rs 7 w 8 9 // 10 // 11 // 12 0 13 rs 14 15 // 16 // 17 // 18 // 19 20 0 21 0 22 0 23 0 24	Output data going out from each vector unit to the SX block content [127:0] sp_sx_data; gg [127:0] osp_sx_data; gg [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult2, VectorResult4, VectorResult4, VectorResult2, VectorResult4, VectorResult4, VectorResult2, VectorResult4, V	ks /ectorResult3; /// // // // // // // // // // // //
2 input [1:0] sq.sp.gpr_phase_mux; 3 input [3:0] sq.sp_channel_mask; 4 input [3:0] sq.sp_channel_mask; 5 input [1:0] sq.sp_gpr_input_mux; 6 input [0:0] sq.sp_mem_rd_ena; 7 input [0:0] sq.sp_mem_wr_ena; 8 input [0:0] sq.sp_mem_wr_ena; 9	nt,sq_sp_constant; ta comming from the Texture Pipe/ Fetch Engine tion address into gpr(s) for texture return data unel mask for the texture data being written into gpr(s) pr_dst, q2_tp_gpr_dst; p_gpr_cmask, q2_tp_gpr_cmask; p_data_valid, q2_tp_data_valid; Page 3 of 22	2 // 3 // 3 // 4 // 5 0 0 6 rs 7 w 8 9 // 10 // 11 // 12 0 13 rs 14 15 // 16 // 17 // 18 // 19 20 0 21 0 22 0 23 0 24	Output data going out from each vector unit to the SX block content [127:0] sp_sx_data; gg [127:0] osp_sx_data; gg [127:0] VectorResult0, VectorResult1, VectorResult2, VectorResult2, VectorResult2, VectorResult3, VectorResult3, VectorResult4, VectorResult2, VectorResult4, VectorResult4, VectorResult2, VectorResult4, VectorResult4, VectorResult2, VectorResult4, V	// vectorResult3;///////////

ATI 2104 LG v. ATI IPR2015-00326

1 //There are four macc units present in a vector unit	1	
2 //The macc units are phased out by one cycle from each other in their execution sequence 3 ///	2 //	
	3 // This data comes from the Scalar Unit	
5tm; 12 2	5 wire [127:0] ScalarData;	
5 reg [0:0] q0_instruct_stall, q1_instruct_stall, q2_instruct_stall; 6 reg [20:0] q0_instruct, q1_instruct, q2_instruct;	6 reg [127:0] q0_ScalarData, q1_ScalarData, q2_ScalarData;	
7 reg [6:0] q0_gpr_wr_addr,q1_gpr_wr_addr,q2_gpr_wr_addr;	7 wire [31:0] ScalarResult;	
8 reg [6:0] q0_gpr_rd_addr,q1_gpr_rd_addr;	8	
9 reg [0:0] q0_gpr_mre,q1_gpr_mre,q2_gpr_mre;	9 //	
0 reg [0:0] q0_gpr_mwe,q1_gpr_mwe,q2_gpr_mwe;	10 wire [31:0] ScalarInput0, ScalarInput1, ScalarInput2, ScalarInput3;	
1 reg [0:0] q0_gpr_we,q1_gpr_we,q2_gpr_we;	11 reg [31:0] ScalarInput;	
2 reg [1:0] q0_gpr_phase_mux,q1_gpr_phase_mux,q2_gpr_phase_mux;	12 reg [5:0] ScalarOpcode;	
<pre>3 reg[1:0] q0_gpr_input_mux,q1_gpr_input_mux,q2_gpr_input_mux;</pre>	13 wire [5:0] ScalarOpcode0, ScalarOpcode1, ScalarOpcode2, ScalarOpcode3;	
4 reg [3:0] q0_gpr_cmask, q1_gpr_cmask, q2_gpr_cmask;	14	
5	15 //	
6 //	16 //daizy-chaining the above buses	
7 //The outputs of the mux unit selecting between iIntepolated, iAutoCount and iVertexIndices	17 //	
8 //	18	
9	19 always@(posedge sclk)	
0 reg [127:0] InputData0, InputData1, InputData2,InputData3;	20 begin	
1 2 //	21 if(srst) 22 begin	
3 //GPR data coming from four different MACC GPR units of the same vector unit	23 q0_instruct_start <= 1'b0;	
4 //	24 q1_instruct_start <= 1'b0;	
5 wire [127:0] RegData0, RegData1, RegData2, RegData3;	25 q2_instruct_start <= 1"b0;	
Page 5 of 22	Prop. 6 of 22	
Page 5 of 22 Ex. 2104 - vector.v	Page 6 of 22 Ex. 2104 - vector.v	
	EX. 2104 - Vector. V	
I of instruct stall <= 150·		
1 q0_instruct_stall <= 1'b0; 2	l ql_gpr_input_mux <= 2'b0;	
2 q1_instruct_stall <= 1'b0;	1 q1_gpr_input_mux <= 2'b0; 2 q2_gpr_input_mux <= 2'b0;	
2	1 q1_gpr_imput_mux <= 2'b0; 2 q2_gpr_input_mux <= 2'b0;	
2	1 q1_gpr_input_mux <= 2'b0; 2 q2_gpr_input_mux <= 2'b0; 3 end // if (srst)	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
2	1	
1	1	
2	1	
1	1	
1	1	
Q_instruct_stall <= 1'b0; Q_instruct_stall <= 1'b0; Q_instruct <= 21'b0; Q_instruct <= 1'b0; Q_instruct <= 7'b0; Q_instruct <= 7'b0; Q_instruct <= 1'b0; Q	1	

```
q2_gpr_mwe <= q1_gpr_mwe;
 2
                q0\_gpr\_we \mathrel{<=} sq\_sp\_wr\_ena;
                q1\_gpr\_we \mathrel{<=} q0\_gpr\_we;
                                                                                                                                   //Muxing logic to select from data comming from the Interpolators(in reality more than just interpolated data....there can be
                q2\_gpr\_we \le q1\_gpr\_we;
                q0\_gpr\_phase\_mux <= sq\_sp\_gpr\_phase\_mux;
                                                                                                                                      //also faceness and XY data), AutoCount data and Vertex Indices comming from the staging
                q1\_gpr\_phase\_mux <= q0\_gpr\_phase\_mux;
                                                                                                                                      //Each MACC unit has its own mux logic since the controls are phased out by one cycle from
                q2\_gpr\_phase\_mux <= q1\_gpr\_phase\_mux;
                                                                                                                                     one MACC to the other
               q0\_gpr\_input\_mux <= sq\_sp\_gpr\_input\_mux;
                                                                                                                              10
11
                q1\_gpr\_input\_mux <= q0\_gpr\_input\_mux;
                                                                                                                              12
                                                                                                                                      //muxing logic for the inputs of the first MACC
                q2\_gpr\_input\_mux <= q1\_gpr\_input\_mux;
                                                                                                                              13
                                                                                                                                      always @(/*AUTOSENSE*/iAutoCount or iInterpolated or iVertexIndices
11
                q0_gpr_cmask <= sq_sp_channel_mask;
                                                                                                                              14
                                                                                                                                              or sq_sp_gpr_input_mux)
12
                q1\_gpr\_cmask \ <= \ q0\_gpr\_cmask;
                                                                                                                              15
13
                q2_gpr_cmask <= q1_gpr_cmask;
                                                                                                                                           case(sq_sp_gpr_input_mux)
14
                q0 tp gpr cmask <= tp sp gpr cmask;
15
                q1_tp_gpr_cmask <= q0_tp_gpr_cmask;
                                                                                                                                             2'b01: InputData0 = iInterpolated;
16
                q2_tp_gpr_cmask <= q1_tp_gpr_cmask;
                                                                                                                              19
                                                                                                                                             2'b10: InputData0 = iVertexIndices;
17
                q0\_tp\_gpr\_dst \, \mathrel{<=} tp\_sp\_gpr\_dst;
                                                                                                                              20
                                                                                                                                             default: InputData0 = iInterpolated;
18
                q1_tp_gpr_dst \le q0_tp_gpr_dst;
                                                                                                                              21
                                                                                                                                           endcase // case(sq_sp_gpr_input_mux)
19
                q2_tp_gpr_dst \le q1_tp_gpr_dst;
                                                                                                                              22
20
                q0\_tp\_data\_valid \mathrel{<=} tp\_sp\_data\_valid;
                                                                                                                              23
21
                q1\_tp\_data\_valid <= q0\_tp\_data\_valid;
                                                                                                                              24
                                                                                                                                      //muxing logic for the inputs of the second MACC
22
                q2\_tp\_data\_valid <= q1\_tp\_data\_valid;
                                                                                                                              25
                                                                                                                                      always @(/*AUTOSENSE*/iAutoCount or iInterpolated or iVertexIndices
23
              end // else: !if(srst)
                                                                                                                              26
                                                                                                                                              or q0_gpr_input_mux)
24
         end // always@ (posedge sclk)
                                                                                                                              2.7
                                                                                                                                       begin
25
                                              Page 9 of 22
                                                                                                                                                                            Page 10 of 22
                                                                                Ex. 2104 - vector.v
                                                                                                                                                                                                              Ex. 2104 - vector.v
                                                                                                                                            2'b00: InputData3 = iAutoCount;
             case(q0 gpr input mux)
 2
              2'b00: InputData1 = iAutoCount;
                                                                                                                                            2'b01: InputData3 = iInterpolated;
              2'b01: InputData1 = iInterpolated;
                                                                                                                                             2'b10: InputData3 = iVertexIndices
              2'b10: InputData1 = iVertexIndices :
                                                                                                                                             default: InputData3 = iInterpolated:
              default: InputData1 = iInterpolated;
                                                                                                                                           endcase // case(q2_gpr_input_mux)
             endcase // case(q0_gpr_input_mux)
 7
                                                                                                                               7
       //muxing logic for the inputs of the third MACC
                                                                                                                              10
                                                                                                                                     //muxing logic for the scalar input into the scalar unit
10
       always @(/*AUTOSENSE*/iAutoCount or iInterpolated or iVertexIndices
                                                                                                                                      //there's one scalar unit for each vector unit in the shader pipe
11
               or q1_gpr_input_mux)
                                                                                                                                     //alpha channel of SrcC is used as input argument into the scalar unit only in cases of an coissued instruction.
12
13
             case(q1_gpr_input_mux)
                                                                                                                              14
15
14
              2'b00: InputData2 = iAutoCount;
                                                                                                                              16
                                                                                                                                      always @(/*AUTOSENSE*/ScalarInput0 or ScalarInput1 or ScalarInput2
15
              2'b01: InputData2 = iInterpolated;
                                                                                                                              17
                                                                                                                                              or ScalarInput3 or sq_sp_gpr_phase_mux)
16
              2'b10: InputData2 = iVertexIndices;
                                                                                                                              18
                                                                                                                                       begin
17
              default: InputData2 = iInterpolated;
                                                                                                                              19
                                                                                                                                           case(sq\_sp\_gpr\_phase\_mux)
18
             endcase // case(q1_gpr_input_mux)
                                                                                                                              20
                                                                                                                                            2'b00: ScalarInput = ScalarInput0;
19
                                                                                                                              21
                                                                                                                                            2'b01: ScalarInput = ScalarInput1;
20
                                                                                                                              22
                                                                                                                                            2'b10: ScalarInput = ScalarInput2;
21
       //muxing logic for the inputs of the fourth MACC
                                                                                                                              23
                                                                                                                                             2'b11: ScalarInput = ScalarInput3;
22
       always @(/*AUTOSENSE*/iAutoCount or iInterpolated or iVertexIndices
                                                                                                                              24
                                                                                                                                             default: ScalarInput = 2'bxx;
23
               or q2_gpr_input_mux)
                                                                                                                              25
                                                                                                                                           endcase // case(sq_sp_gpr_phase_mux)
24
         begin
25
             case(q2_gpr_input_mux)
                                              Page 11 of 22
                                                                                                                                                                            Page 12 of 22
                                                                               Ex. 2104 - vector.v
                                                                                                                                                                                                              Ex. 2104 - vector.v
```

```
2
              always @(/*AUTOSENSE*/ScalarOpcode0 or ScalarOpcode1
                                                                                                                                                                                                                                              //Instantiation of all four MACC units that create a Vector Unit
 3
                           or ScalarOpcode2 or ScalarOpcode3 or sq_sp_gpr_phase_mux)
 4
                       case(sq_sp_gpr_phase_mux)
                                                                                                                                                                                                                                            umacc gpr0(.oVectorOutput(VectorResult0),.oScalarInput(ScalarInput0),.oScalarOpcode(Scala
                        2'b00: ScalarOpcode = ScalarOpcode0;
                                                                                                                                                                                                                                            rOpcode 0), o Reg Data (Reg Data 0), o export\_dst (sq\_sp\_exp\_dst), \\
                         2'b01: ScalarOpcode = ScalarOpcode1;
                                                                                                                                                                                                                                            . sq\_sp\_instruct(sq\_sp\_instruct),. sq\_sp\_instruct\_start(sq\_sp\_instruct\_start),. sq\_sp\_gpr\_rd\_addr(sq\_sp\_gpr\_rd\_addr),
                         2'b10: ScalarOpcode = ScalarOpcode2;
                         2'b11: ScalarOpcode = ScalarOpcode3;
                                                                                                                                                                                                                                           .sq_sp_gpr_wr_addr(sq_sp_wr_addr),.sq_sp_wr_ena(sq_sp_wr_ena),.sq_sp_mem_rd_ena(sq_s p_mem_rd_ena),.sq_sp_mem_wr_ena(sq_sp_mem_wr_ena),
                         default: ScalarOpcode = 2'bxx;
11
                       endcase // case(sq_sp_gpr_phase_mux)
                                                                                                                                                                                                                                 15
                                                                                                                                                                                                                                                                             .sq sp gpr cmask(sq sp channel mask),
12
                                                                                                                                                                                                                                            . sq\_sp\_gpr\_phase\_mux(sq\_sp\_gpr\_phase\_mux), iInterpolated(InputData0),.sq\_sp\_constant(sq\_sp\_constant), \\
13
14
             //module scalar_lut( iAG_ME_OPCODE, iAG_ME_IN_A, iAG_ME_IN_B, iAG_ME_IN_C,
                                                                                                                                                                                                                                 19
                                                                                                                                                                                                                                                                             . iS calar Data (Scalar Data), .tp\_sp\_data (tp\_sp\_data),
           /\!/ \, iAG_ME_ABS_A, iAG_ME_ABS_B, iAG_ME_ABS_C, iAG_ME_A_NEGATE, iAG_ME_B_NEGATE
 15
16
                                                                                                                                                                                                                                            .tp\_sp\_gpr\_dst(tp\_sp\_gpr\_dst),\\.tp\_sp\_gpr\_cmask(tp\_sp\_gpr\_cmask),.tp\_sp\_data\_valid(tp\_sp\_data\_valid),\\
17
              //, iAG_ME_C_NEGATE, oME_RESULT, sclk );
                                                                                                                                                                                                                                 22
                                                                                                                                                                                                                                                                           .sclk(sclk), .srst(srst));
18
                                                                                                                                                                                                                                 23
19
                                                                                                                                                                                                                                            macc_gpr[i(.oVectorOutput(VectorResult1),.oScalarInput(ScalarInput1),.oScalarOpcode(ScalarOpcode1),.oRegData(RegData1),.sq_sp_instruct(q0_instruct),.sq_sp_instruct_start(q0_instruct_start),.sq_sp_gpr_rd_addr(q0_gpr_rd_addr),
20
             //Scalar Unit instantiation
21
                                                                                                                                                                                                                                 27
          scalar\_lut \qquad uscalar(ScalarOpcode,ScalarInput, \\ 1'b0,1'b0,1'b0,1'b0,1'b0,1'b0,ScalarResult,sclk);
                                                                                                                                      ScalarInput,ScalarInput,
                                                                                                                                                                                                                                           .sq\_sp\_gpr\_wr\_addr(q0\_gpr\_wr\_addr),.sq\_sp\_wr\_ena(q0\_gpr\_we),.sq\_sp\_mem\_rd\_ena(q0\_gpr\_mre),.sq\_sp\_mem\_wr\_ena(q0\_gpr\_mwe),
24
                                                                                                                                                                                                                                                                             .sq_sp_gpr_cmask(q0_gpr_cmask),
25
                                                                                   Page 13 of 22
                                                                                                                                                                                                                                                                                                                   Page 14 of 22
                                                                                                                                              Ex. 2104 - vector.v
                                                                                                                                                                                                                                                                                                                                                                                Ex. 2104 - vector.v
           .sq\_sp\_gpr\_phase\_mux(q0\_gpr\_phase\_mux),. iInterpolated(InputData1),.sq\_sp\_constant(sq\_sp\_constant), \\
                                                                                                                                                                                                                                            .sq\_sp\_gpr\_phase\_mux(q2\_gpr\_phase\_mux),. iInterpolated(InputData3),.sq\_sp\_constant(sq\_sp\_constant), and the properties of the properties
                                             . iScalar Data (q0\_Scalar Data), .tp\_sp\_data (tp\_sp\_data), \\
                                                                                                                                                                                                                                   4
                                                                                                                                                                                                                                                                              . iScalar Data (q2\_Scalar Data), .tp\_sp\_data (tp\_sp\_data), .sclk (sclk), \\
            .tp\_sp\_gpr\_dst(q0\_tp\_gpr\_dst),\\ .tp\_sp\_gpr\_cmask(q0\_tp\_gpr\_cmask),.tp\_sp\_data\_valid(q0\_tp\_data\_valid),\\
                                                                                                                                                                                                                                            .tp\_sp\_gpr\_dst(q2\_tp\_gpr\_dst), \\ .tp\_sp\_gpr\_cmask(q2\_tp\_gpr\_cmask), .tp\_sp\_data\_valid(q2\_tp\_data\_valid), \\
          macc_gpr
umacc_gpr2(_oVectorOutput(VectorResult2)_,oScalarInput(ScalarInput2)_,oScalarOpcode(ScalarOpcode2)_,oRegData(RegData2)_,sq_sp_instruct(q1_instruct)_,sq_sp_instruct_start(q1_instruct_start)_,sq_sp_gpr_rd_addr(q1_gpr_rd_addr)_
                                                                                                                                                                                                                                                //Muxing the gpr vector results into one final vector result conrolled by the phase mux signal
                                                                                                                                                                                                                                 11
                                                                                                                                                                                                                                            or a registered version of it
13
                                                                                                                                                                                                                                 13
          .sq\_sp\_gpr\_wr\_addr(q1\_gpr\_wr\_addr), sq\_sp\_wr\_ena(q1\_gpr\_we), .sq\_sp\_mem\_rd\_ena(q1\_gpr\_mwe), .sq\_sp\_mem\_wr\_ena(q1\_gpr\_mwe), \\
                                                                                                                                                                                                                                 14
16
                                                                                                                                                                                                                                 15
                                                                                                                                                                                                                                                always @(/*AUTOSENSE*/VectorResult0 or VectorResult1
                                            .sq\_sp\_gpr\_cmask(q1\_gpr\_cmask),
                                                                                                                                                                                                                                 16
                                                                                                                                                                                                                                                            or VectorResult2 or VectorResult3 or sq_sp_gpr_phase_mux)
            17
19
                                                                                                                                                                                                                                 18
                                                                                                                                                                                                                                                         case(sq_sp_gpr_phase_mux)
20
                                            . iS calar Data (q1\_Scalar Data), .tp\_sp\_data (tp\_sp\_data),
                                                                                                                                                                                                                                 19
                                                                                                                                                                                                                                                           2'b00: osp_sx_data = VectorResult0;
            .tp\_sp\_gpr\_dst(q1\_tp\_gpr\_dst),\\ .tp\_sp\_gpr\_cmask(q1\_tp\_gpr\_cmask),.tp\_sp\_data\_valid(q1\_tp\_data\_valid),\\
21
22
                                                                                                                                                                                                                                 20
                                                                                                                                                                                                                                                           2'b01: osp_sx_data = VectorResult1;
23
                                            .sclk(sclk), .srst(srst));
                                                                                                                                                                                                                                                           2'b10: osp_sx_data = VectorResult2;
24
                                                                                                                                                                                                                                 22
                                                                                                                                                                                                                                                           2'b11: osp_sx_data = VectorResult3;
25
26
27
28
                                                                                                                                                                                                                                 23
          nnacc_gpr3(,oVectorOutput(VectorResult3),oScalarInput(ScalarInput3),oScalarOpcode(ScalarOpcode),oRegData(RegData3),sq_sp_instruct(q2_instruct),sq_sp_instruct_start(q2_instruct_start),sq_sp_gpr_rd_addr(q2_gpr_rd_addr),
                                                                                                                                                                                                                                                         endcase // case(sq_sp_gpr_phase_mux)
                                                                                                                                                                                                                                24
                                                                                                                                                                                                                                 25
            .sq_sp_gpr_wr_addr(q2_gpr_wr_addr),.sq_sp_wr_ena(q2_gpr_we),.sq_sp_mem_rd_ena(q2_gpr_mre),.sq_sp_mem_wr_ena(q2_gpr_mwe),
30
31
                                                                                                                                                                                                                                 26
                                                                                                                                                                                                                                               assign sp sx data = osp sx data;
                                                                                                                                                                                                                                 27
32
                                           .sq_sp_gpr_cmask(q2_gpr_cmask),
                                                                                  Page 15 of 22
                                                                                                                                                                                                                                                                                                                   Page 16 of 22
                                                                                                                                              Ex. 2104 - vector.v
                                                                                                                                                                                                                                                                                                                                                                                Ex. 2104 - vector.v
```

```
o0 ScalarData <= ScalarData:
                                                                                                                                         q1 ScalarData <= q0 ScalarData;
       //Muxing the gpr outputs (texture data) into one 128 bit bus going out to the texture unit
                                                                                                                                         q2 ScalarData <= q1 ScalarData;
        always @(/*AUTOSENSE*/RegData0 or RegData1 or RegData2 or RegData3
              or sq sp gpr phase mux)
                                                                                                                                   //simple pipelining to delay the sq to sx control signals (via sp)
        begin
                                                                                                                                         q0_sq_exp_pvalid,q1_sq_exp_pvalid,q2_sq_exp_pvalid,q3_sq_exp_pvalid,q4_sq_exp_
            case(sq sp gpr phase mux)
10
             2'b00: osp tp data = RegData0[95:0];
                                                                                                                                         q5_sq_exp_pvalid,q6_sq_exp_pvalid,q7_sq_exp_pvalid,q8_sq_exp_pvalid,q9_sq_exp_
                                                                                                                            11
12
11
              2'b01: osp tp data = RegData1[95:0];
                                                                                                                                  pvalid:
12
             2'b10: osp tp data = RegData2[95:0];
                                                                                                                            13
                                                                                                                                   reg~[3:0] \qquad q10\_sq\_exp\_pvalid, q11\_sq\_exp\_pvalid;
13
             2'b11: osp tp data = RegData3[95:0];
                                                                                                                            14
14
            endcase // case(sq_sp_gpr_phase_mux)
                                                                                                                                         q0_sq_exporting,q1_sq_exporting,q2_sq_exporting,q3_sq_exporting,q4_sq_exporting;
15
16
                                                                                                                                         q5_sq_exporting,q6_sq_exporting,q7_sq_exporting,q8_sq_exporting,q9_sq_exporting;
17
        assign sp_tp_data = osp_tp_data;
                                                                                                                            19
                                                                                                                                   reg [0:0] q10_sq_exporting,q11_sq_exporting;
18
                                                                                                                            20
19
                                                                                                                                         q0_sq_exp_alu_id,q1_sq_exp_alu_id,q2_sq_exp_alu_id,q3_sq_exp_alu_id,q4_sq_exp_
20
       //replicate the scalar result into all four channels so that it can be used as an 128-bit input
                                                                                                                            23
21
        //into the vector unit argument selection logic as PS (Previous Scalar Result)
                                                                                                                            24
25
26
                                                                                                                                         q5_sq_exp_alu_id,q6_sq_exp_alu_id,q7_sq_exp_alu_id,q8_sq_exp_alu_id,q9_sq_exp_
22
23
        assign\ ScalarData = \{ScalarResult, ScalarResult, ScalarResult\};\\
                                                                                                                            27
                                                                                                                                   reg [0:0] q10_sq_exp_alu_id,q11_sq_exp_alu_id;
24
                                                                                                                            28
25
        always @(posedge sclk)
                                                                                                                                   reg [5:0] q0 sq exp dst,q1 sq exp dst,q2 sq exp dst,q3 sq exp dst,q4 sq exp dst;
        begin
                                                                                                                                   reg~[5:0] \qquad q5\_sq\_exp\_dst, q6\_sq\_exp\_dst, q7\_sq\_exp\_dst, q8\_sq\_exp\_dst, q9\_sq\_exp\_dst; \\
                                             Page 17 of 22
                                                                                                                                                                         Page 18 of 22
                                                                              Ex. 2104 - vector.v
                                                                                                                                                                                                          Ex. 2104 - vector.v
                                                                                                                                            q3_sq_exp_pvalid <= q2_sq_exp_pvalid;
       reg [5:0] q10 sq exp dst,q11 sq exp dst;
 2
                                                                                                                                            q4 sq exp pvalid <= q3 sq exp pvalid;
                                                                                                                                            q5_sq_exp_pvalid <= q4_sq_exp_pvalid;
 4
                                                                                                                                            q6\_sq\_exp\_pvalid <= q5\_sq\_exp\_pvalid;
       always@(posedge sclk)
        begin
                                                                                                                                            q7_sq_exp_pvalid <= q6_sq_exp_pvalid;
            if(srst)
                                                                                                                                            q8_sq_exp_pvalid <= q7_sq_exp_pvalid;
                                                                                                                                            q9\_sq\_exp\_pvalid <= q8\_sq\_exp\_pvalid;
               q0\_sq\_exporting <= 2\text{'b}00;
                                                                                                                                            q10\_sq\_exp\_pvalid <= q9\_sq\_exp\_pvalid;
               q1_sq_exporting <= 2'b00;
                                                                                                                                            q11\_sq\_exp\_pvalid <= q10\_sq\_exp\_pvalid;
10
               q2\_sq\_exporting <= 2"b00;
                                                                                                                            10
11
               q3_sq_exporting <= 2'b00;
                                                                                                                            11
                                                                                                                                            q0\_sq\_exporting <= sq\_sp\_exporting;
               q4\_sq\_exporting <= 2"b00";
12
                                                                                                                            12
                                                                                                                                            q1\_sq\_exporting <= q0\_sq\_exporting;
13
               q5_sq_exporting <= 2'b00;
                                                                                                                                            q2\_sq\_exporting <= q1\_sq\_exporting;
               q6_sq_exporting <= 2'b00;
                                                                                                                                            q3_sq_exporting <= q2_sq_exporting;
               q7_sq_exporting <= 2'b00;
                                                                                                                                            q4_sq_exporting <= q3_sq_exporting;
               q8_sq_exporting <= 2'b00;
                                                                                                                                            q5_sq_exporting <= q4_sq_exporting;
               q9_sq_exporting <= 2'b00;
                                                                                                                                            q6_sq_exporting <= q5_sq_exporting;
               q10 sq exporting <= 2'b00;
                                                                                                                                            q7 sq exporting <= q6 sq exporting;
               q11_sq_exporting <= 2'b00;
                                                                                                                                            q8 sq exporting <= q7 sq exporting;
20
              end
                                                                                                                            20
                                                                                                                                            q9 sq exporting <= q8 sq exporting;
21
             else
                                                                                                                                            q10 sq exporting <= q9 sq exporting;
22
                                                                                                                            22
                                                                                                                                            q11_sq_exporting <= q10_sq_exporting;
23
                                                                                                                            23
               q0_sq_exp_pvalid <= sq_sp_exp_pvalid;
24
               q1_sq_exp_pvalid <= q0_sq_exp_pvalid;
                                                                                                                            24
                                                                                                                                            q0_sq_exp_alu_id <= sq_sp_exp_alu_id;
               q2\_sq\_exp\_pvalid <= q1\_sq\_exp\_pvalid;
                                                                                                                            25
                                                                                                                                            q1\_sq\_exp\_alu\_id \mathrel{<=} q0\_sq\_exp\_alu\_id;
                                             Page 19 of 22
                                                                                                                                                                         Page 20 of 22
                                                                              Ex. 2104 - vector.v
                                                                                                                                                                                                          Ex. 2104 - vector.v
```

```
1
                 q2_sq_exp_alu_id <= q1_sq_exp_alu_id;
                                                                                                                                                 1
                                                                                                                                                 2
 2
                 q3\_sq\_exp\_alu\_id \mathrel{<=} q2\_sq\_exp\_alu\_id;
                 q4_sq_exp_alu_id <= q3_sq_exp_alu_id;
                                                                                                                                                        //assigning the outputs
                 q5\_sq\_exp\_alu\_id \mathrel{<=} q4\_sq\_exp\_alu\_id;
                                                                                                                                                         assign\ sp\_sx\_exp\_pvalid\ = q11\_sq\_exp\_pvalid;
                  q6\_sq\_exp\_alu\_id <= q5\_sq\_exp\_alu\_id;
                                                                                                                                                         assign\ sp\_sx\_exp\_alu\_id\ = q11\_sq\_exp\_alu\_id;
                 q7\_sq\_exp\_alu\_id \mathrel{<=} q6\_sq\_exp\_alu\_id;
                                                                                                                                                         assign\ sp\_sx\_exporting\ = q11\_sq\_exporting;
                  q8\_sq\_exp\_alu\_id \mathrel{<=} q7\_sq\_exp\_alu\_id;
                                                                                                                                                         assign\ sp\_sx\_exp\_dst\ = q10\_sq\_exp\_dst;
                 q9_sq_exp_alu_id <= q8_sq_exp_alu_id;
                  q10\_sq\_exp\_alu\_id \mathrel{<=} q9\_sq\_exp\_alu\_id;
                                                                                                                                                 9 endmodule // vector
10
                  q11\_sq\_exp\_alu\_id \mathrel{<=} q10\_sq\_exp\_alu\_id;
                                                                                                                                                10
11
                                                                                                                                                11
12
                  q0\_sq\_exp\_dst \mathrel{<=} sq\_sp\_exp\_dst;
                                                                                                                                                12
13
                 q1\_sq\_exp\_dst \mathrel{<=} q0\_sq\_exp\_dst;
                                                                                                                                                 13
14
                 q2\_sq\_exp\_dst \mathrel{<=} q1\_sq\_exp\_dst;
                                                                                                                                                14
15
                  q3\_sq\_exp\_dst <= q2\_sq\_exp\_dst;
16
                 q4\_sq\_exp\_dst \mathrel{<=} q3\_sq\_exp\_dst;
17
                 q5\_sq\_exp\_dst \mathrel{<=} q4\_sq\_exp\_dst;
                 q6\_sq\_exp\_dst \le q5\_sq\_exp\_dst;
19
                 q7\_sq\_exp\_dst \le q6\_sq\_exp\_dst;
20
                  q8\_sq\_exp\_dst \mathrel{<=} q7\_sq\_exp\_dst;
21
                  q9\_sq\_exp\_dst \le q8\_sq\_exp\_dst;
22
                 q10\_sq\_exp\_dst \mathrel{<=} q9\_sq\_exp\_dst;
23
                 q11\_sq\_exp\_dst \mathrel{<=} q10\_sq\_exp\_dst;
24
                end // else: !if(srst)
25
          end // always@ (posedge sclk)
                                                     Page 21 of 22
                                                                                                                                                                                                     Page 22 of 22
                                                                                           Ex. 2104 - vector.v
                                                                                                                                                                                                                                            Ex. 2104 - vector.v
```

```
1 //
                       -*- Mode: Verilog -*-
                                                                                                                           1 //i/o declaration
 2 // Filename : macc reg.v
                                                                                                                           2 //For the signals coming straight from the top level without going through
 3 // Description : This module represents the MACC (Multiply and Accumulate) unit plus
                                                                                                                                 //any combinational logic...the signal name is preserved...but in lower case
              : the corrensponding GPR (register file) module.
 5 // Author : Andi Skende
                                                                                                                                input [20:0] sq_sp_instruct;
 6 // Created On : Fri Feb 1 15:53:03 2002
                                                                                                                           6 input [0:0] sq_sp_instruct_start;
 7 // Last Modified By: .
                                                                                                                           7 input [6:0] sq_sp_gpr_rd_addr,sq_sp_gpr_wr_addr;
 8 // Last Modified On:
                                                                                                                                 input [1:0] sq_sp_gpr_phase_mux;
                                                                                                                                 input [0:0] sq_sp_mem_wr_ena,sq_sp_mem_rd_ena,sq_sp_wr_ena;
10 // Status : Unknown, Use with caution!
                                                                                                                                 input [3:0] sq_sp_gpr_cmask;
                                                                                                                                input [127:0] iInterpolated ,sq_sp_constant,iScalarData,tp_sp_data;
11
                                                                                                                          11
                                                                                                                          12 input [6:0] tp_sp_gpr_dst;
13 module macc_gpr(
                                                                                                                          13
                                                                                                                                input [3:0] tp_sp_gpr_cmask;
14
                  /*AUTOARG*/
                                                                                                                          14
                                                                                                                                input [0:0] tp_sp_data_valid;
15
                                                                                                                          15
                                                                                                                                 input [0:0] sclk,srst;
16
       oScalar Input, oScalar Opcode, oVector Output, oReg Data, oexport\_dst,
17 // Inputs
                                                                                                                          17
                                                                                                                                 output [31:0] oScalarInput;
18 sq sp instruct, sq sp instruct start, sq sp gpr rd addr,
                                                                                                                          18
                                                                                                                                output [5:0] oScalarOpcode;
19 sq_sp_gpr_wr_addr, sq_sp_gpr_phase_mux, sq_sp_mem_wr_ena,
                                                                                                                                 output [127:0] oVectorOutput;
20 sq_sp_mem_rd_ena, sq_sp_wr_ena, sq_sp_gpr_cmask, iInterpolated,
                                                                                                                          20
                                                                                                                                output [127:0] oRegData; //data coming out of the register files
21 sq_sp_constant, iScalarData, tp_sp_data, tp_sp_gpr_dst,
                                                                                                                          21
                                                                                                                                 output [5:0] oexport dst;
       tp_sp_gpr_cmask, tp_sp_data_valid, sclk, srst
23
                                                                                                                          23
                                                                                                                                 wire [127:0] RegData; //data coming out of the register files
                                                                                                                                wire [127:0] VectorResult;
24
                                                                                                                                reg [127:0] InputGPR;
                                             Page 1 of 16
                                                                                                                                                                       Page 2 of 16
                                                                          Ex. 2105 - macc gpr.v
                                                                                                                                                                                                    Ex. 2105 - macc gpr.v
                                                                                                                                 //----
                                                                                                                           2 reg [6:0] gpr_wr_addr;
                                                                                                                                reg [3:0] gpr_wr_mask;
 4 //Instantiation of the mace unit which does the argument selection and input modification 5 (swizzling ...etc)
       //1. input for the scalar unit comes as an output from this unit and goes all the way up to vector.v module where the instance of scalar unit
                                                                                                                                 always@(/*AUTOSENSE*/sq_sp_gpr_phase_mux or sq_sp_gpr_wr_addr
                                                                                                                                       or tp_sp_gpr_dst)
 8 // can be found.
9 //2. VectorResult output is only used as an input into GPRs .... the Previous Vector Result is
10 not exposed at this level but stays internal
                                                                                                                           8
                                                                                                                                      case(sq_sp_gpr_phase_mux)
11 // to mace v module
                                                                                                                          10
                                                                                                                          11
                                                                                                                                       2'b11: gpr_wr_addr = sq_sp_gpr_wr_addr;
                                                                                                                          12
                                                                                                                                       2'b01: gpr wr addr = tp sp gpr dst;
15
       reg [127:0] q_RegData;
                                                                                                                          13
                                                                                                                                      endcase // case(sq_sp_gpr_phase_mux)
16
                                                                                                                          14
                                                                                                                                  end // always@ (...
                                                                                                                          15
      \label{lem:condition} \begin{split} & macc(oResult(VectorResult), oScalarOpcode(oScalarOpcode), oScalarInput(oScalarInput), o \\ & ExportDst(oexport\_dst), \end{split}
                                                                                                                                  always@(/*AUTOSENSE*/sq_sp_gpr_cmask or sq_sp_gpr_phase_mux
             . i Reg Data (q\_Reg Data), i Constant Data (sq\_sp\_constant), i Scalar Data (i Scalar Data), \\
                                                                                                                          17
                                                                                                                                       or tp_sp_gpr_cmask)
21
                .iInstruction(sq_sp_instruct), .iInstStart(sq_sp_instruct_start), .sclk(sclk), .srst(srst));
                                                                                                                          18
                                                                                                                                  begin
                                                                                                                                      case(sq_sp_gpr_phase_mux)
23
                                                                                                                          20
                                                                                                                                       2'b00.
24
                                                                                                                          21
                                                                                                                                       2'b10.
                                                                                                                                       2"b11: gpr_wr_mask = sq_sp_gpr_cmask;
26\, \, //We need to mask between the write controls coming from the TP and SQ \,
                                                                                                                          23
27 //TP sends its own destination pointer and channel mask
                                                                                                                          24
                                                                                                                                      endcase \textit{//} case(sq\_sp\_gpr\_phase\_mux)
       //We also get a destination pointer and channel mask from SQ
                                                                                                                               end // always@ (...
                                                                                                                                                                      Page 4 of 16
                                             Page 3 of 16
                                                                                                                                                                                                    Ex. 2105 - macc gpr.v
                                                                          Ex. 2105 - macc gpr.v
```

ATI 2105 LG v. ATI IPR2015-00326

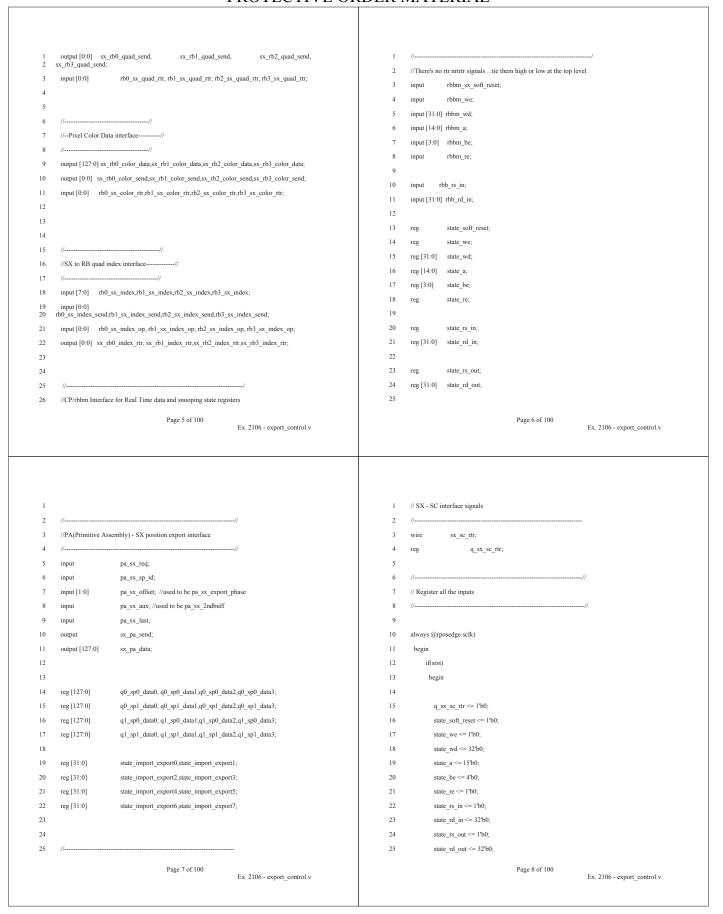
```
2'b00: InputGPR = iInterpolated;
 2
                                                                                                                                                                                                                           2'b01: InputGPR = tp_sp_data;
 3
           wire tp_sp_wr_ena;
                                                                                                                                                                                                                           2'b10: InputGPR = VectorResult:
 4
          assign tp_sp_wr_ena = tp_sp_data_valid;
                                                                                                                                                                                                                           2'b11: InputGPR = iScalarData;
                                                                                                                                                                                                                           default: InputGPR = iInterpolated;
                                                                                                                                                                                                                          endcase // case(sq_sp_gpr_phase_mux)
           always@(/*AUTOSENSE*/sq\_sp\_gpr\_phase\_mux\ or\ sq\_sp\_wr\_ena
                       or tp_sp_wr_ena)
                    case(sq_sp_gpr_phase_mux)
 10
                                                                                                                                                                                                     10
11
                                                                                                                                                                                                     11
                                                                                                                                                                                                                 //Behavioral model of a 128x128 Register File used to emulate GPRs
12
                     2'b11: gpr_wr_ena = sq_sp_wr_ena;
                                                                                                                                                                                                     12
13
                                                                                                                                                                                                     13
                                                                                                                                                                                                               // dum_mem_p2 #(7,128,128) udum_mem(.iRCLK(sclk),
                     2'b01: gpr wr ena = tp sp wr ena;
14
                                                                                                                                                                                                     14
                                                                                                                                                                                                                                                  .iWCLK(sclk),
                    endcase // case(sq_sp_gpr_phase_mux)
15
                                                                                                                                                                                                     15
              end // always@.(...
                                                                                                                                                                                                                                        .iMER(sq_sp_mem_rd_ena),
16
                                                                                                                                                                                                     16
                                                                                                                                                                                                                            .iMEW(sq_sp_mem_wr_ena),
17
                                                                                                                                                                                                     17
                                                                                                                                                                                                               // .iWEN(sq_sp_wr_ena),
                                                                                                                                                                                                     18
                                                                                                                                                                                                                // .iRADR(sq_sp_gpr_rd_addr).
                                                                                                                                                                                                     19
                                                                                                                                                                                                                // .iWADR(sq_sp_gpr_wr_addr),
20
           //The phase mux controlling the write input port into GPRs (register file write port)
                                                                                                                                                                                                     20
                                                                                                                                                                                                                 // .iD(InputGPR).
21
22
                                                                                                                                                                                                    21
                                                                                                                                                                                                                 // .oQ(RegData)
23
                                                                                                                                                                                                     22
            always@(/*AUTOSENSE*/VectorResult or iInterpolated or iScalarData
24
                      or sq_sp_gpr_phase_mux or tp_sp_data)
                                                                                                                                                                                                     23
25
                                                                                                                                                                                                     24
                                                                                                                                                                                                                 wire [127:0] subword_write_mask
              begin
                                                                                                                                                                                                     25
                    case(sq sp gpr phase mux)
                                                                         Page 5 of 16
                                                                                                                                                                                                                                                                              Page 6 of 16
                                                                                                                        Ex. 2105 - macc_gpr.v
                                                                                                                                                                                                                                                                                                                            Ex. 2105 - macc_gpr.v
           //since the virage behavioral model does not give a better mask/pin ration than 1:1
                                                                                                                                                                                                                 rfsd2 128x128cm2sw1 ugpr mem
                                                                                                                                                                                                               (/*VRGIO rfsd2_128x128cm2sw1 InputGPR RegData gpr_wr_addr sq_sp_gpr_rd_addr gpr_wr_ena sq_sp_gpr_rd_ena null*/
 2
           //i am generating the 32-bit-subword mask in the following fashion
                              subword write mask[31:0] = {32{gpr wr mask[0]}};
           assign
                                                                                                                                                                                                                   // READ INTERFACE
                              subword write mask[63:32] = {32{gpr wr mask[1]}};
           assign
                                                                                                                                                                                                                    .CLKB(iSCLK), // Read Clock
           assign
                              subword_write_mask[95:64] = {32{gpr_wr_mask[2]}};
                                                                                                                                                                                                                    . OEB(sq\_sp\_gpr\_rd\_ena), /\!/\ Output\ enable
            assign
                              subword_write_mask[127:96] = {32{gpr_wr_mask[3]}};
                                                                                                                                                                                                                     .MEB(vdd), // Read enable
                                                                                                                                                                                                              . ADRB0(sq\_sp\_gpr\_rd\_addr[0]), \\ . ADRB1(sq\_sp\_gpr\_rd\_addr[1]), \\ . ADRB2(sq\_sp\_gpr\_rd\_addr[2]), \\ . ADRB3(sq\_sp\_gpr\_rd\_addr[3]), \\ // Read\ Address
          'ifdef USE BEHAVE MEM
                                                                                                                                                                                                               . ADRB4(sq\_sp\_gpr\_rd\_addr[4]), \\ . ADRB6(sq\_sp\_gpr\_rd\_addr[6]), \ // \ Read \ Address
                                                                                                                                                                                                                                                                                                            .ADRB5(sq_sp_gpr_rd_addr[5]),
10
            rfsd2_128x128cm1sw8_core ugpr_mem(.QB(RegData),
                                                                                                                                                                                                              .QB0(RegData[0]), .QB1(RegData[1]), .QB2(RegData[2]), .QB3(RegData[3]), \ /\!/ Read Data
11
                                                         .ADRA_buf(gpr_wr_addr),
12
                                                          .DA_buf(InputGPR),
                                                                                                                                                                                                                   . QB4(RegData[4]), \;\; . QB5(RegData[5]), \;\; . QB6(RegData[6]), \;\; . QB7(RegData[7]), \quad // \;\; Read
                                                                                                                                                                                                     15
13
                                                           .WEMA_buf(subword_write_mask),
                                                                                                                                                                                                     16
17
                                                                                                                                                                                                                     . QB8(RegData[8]), \quad . QB9(RegData[9]), \quad . QB10(RegData[10]), \quad . QB11(RegData[11]), \quad // \\
                                                           .WEA_buf(gpr_wr_ena),
                                                                                                                                                                                                              Read Data
                                                           .MEA_buf(gpr_wr_ena),
                                                                                                                                                                                                              .
QB12(RegData[12]), .
QB13(RegData[13]), .
QB14(RegData[14]), .
QB15(RegData[15]), \ensuremath{//} Read Data
                                                           .CLKA(sclk),
                                                                                                                                                                                                     20
21
                                                                                                                                                                                                              .
QB16(RegData[16]), .
QB17(RegData[17]), .
QB18(RegData[18]), .
QB19(RegData[19]), /\!/ Read Data
17
                                                           .BISTEA(1'b0),
18
                                                           .ADRB buf(sq sp gpr rd addr),
                                                                                                                                                                                                                     . QB20 (RegData[20]), \ . QB21 (RegData[21]), \ . QB22 (RegData[22]), \ . QB23 (RegData[23]), \ \# \ . \\
                                                                                                                                                                                                     22
23
                                                           .OEB buf(1'b1),
                                                                                                                                                                                                                    . QB24 (RegData[24]), \ . QB25 (RegData[25]), \ . QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB27 (RegData[27]), \ \# 1000 \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . QB26 (RegData[26]), \ . \\ QB26 (RegData[26]), \ . QB26 (RegD
                                                           .MEB_buf(sq_sp_mem_rd_ena),
20
                                                                                                                                                                                                     25
21
                                                           .CLKB(sclk),
                                                                                                                                                                                                              .
QB28(RegData[28]), .
QB29(RegData[29]), .
QB30(RegData[30]), .
QB31(RegData[31]), // Read Data
                                                                                                                                                                                                     26
27
22
                                                           .BISTEB(1'b0).
23
                                                           .AWTB(1'b0)
                                                                                                                                                                                                               .QB32(RegData[32]), .QB33(RegData[33]), .QB34(RegData[34]), .QB35(RegData[35]), /\!/ Read Data
24
                                                                                                                                                                                                              .QB36(RegData[36]), .QB37(RegData[37]), .QB38(RegData[38]), .QB39(RegData[39]), \ensuremath{//} Read Data
         'else // !'ifdef USE BEHAVE MEM
                                                                         Page 7 of 16
                                                                                                                                                                                                                                                                              Page 8 of 16
                                                                                                                       Ex. 2105 - macc gpr.v
                                                                                                                                                                                                                                                                                                                            Ex. 2105 - macc gpr.v
```

1 .QB40(RegData[40]), .QB41(RegData[41]), .QB42(RegData[42]), .QB43(RegData[43]), // 2 Read Data	1 .QB104(RegData[104]), .QB105(RegData[105]), .QB106(RegData[106]), 2 .QB107(RegData[107]), // Read Data
3 .QB44(RegData[44]), .QB45(RegData[45]), .QB46(RegData[46]), .QB47(RegData[47]), //	3 .QB108(RegData[108]), .QB109(RegData[109]), .QB110(RegData[110]),
4 Read Data 5 .QB48(RegData[48]), .QB49(RegData[49]), .QB50(RegData[50]), .QB51(RegData[51]), //	4 .QB111(RegData[111]), // Read Data 5 .QB112(RegData[112]), .QB113(RegData[113]), .QB114(RegData[114]),
6 Read Data 7 .QB52(RegData[52]), .QB53(RegData[53]), .QB54(RegData[54]), .QB55(RegData[55]), //	6 .QB115(RegData[115]), // Read Data 7 .QB116(RegData[116]), .QB117(RegData[117]), .QB118(RegData[118]),
 Read Data QB56(RegData[56]), QB57(RegData[57]), QB58(RegData[58]), QB59(RegData[59]), // 	8 .QB119(RegData[119]), // Read Data 9 .QB120(RegData[120]), .QB121(RegData[121]), .QB122(RegData[122]),
10 Read Data 11 .QB60(RegData[60]), .QB61(RegData[61]), .QB62(RegData[62]), .QB63(RegData[63]), //	10 .QB123(RegData[123]),
12 Read Data 13 .QB64(RegData[64]), .QB65(RegData[65]), .QB66(RegData[66]), .QB67(RegData[67]), //	12 .QB127(RegData[127]), // Read Data 13 // WRITE INTERFACE
14 Read Data	14 .CLKA(iSCLK), // Write Clock
16 Read Data	15 .WEA(gpr_wr_ena), // Write enable
17 .QB72(RegData[72]), QB73(RegData[73]), .QB74(RegData[74]), .QB75(RegData[75]), // 18 Read Data	16 .MEA(vdd), // Memory enable 17 .ADRA0(gpr_wr_addr[0]), .ADRA1(gpr_wr_addr[1]), .ADRA2(gpr_wr_addr[2]),
19 .QB76(RegData[76]), .QB77(RegData[77]), .QB78(RegData[78]), .QB79(RegData[79]), // 20 Read Data	18 .ADRA3(gpr_wr_addr[3]), // Write Address 19 .ADRA4(gpr_wr_addr[4]), .ADRA5(gpr_wr_addr[5]), .ADRA6(gpr_wr_addr[6]), // Write
21 .QB80(RegData[80]), .QB81(RegData[81]), .QB82(RegData[82]), .QB83(RegData[83]), //	20 Address 21 .DA0(InputGPR[0]), .DA1(InputGPR[1]), .DA2(InputGPR[2]), .DA3(InputGPR[3]), //
23QB84(RegData[84]), .QB85(RegData[85]), .QB86(RegData[86]), .QB87(RegData[87]), $\ //$ 24Read Data	22 Write Data 23 .DA4(InputGPR[4]), .DA5(InputGPR[5]), .DA6(InputGPR[6]), .DA7(InputGPR[7]), //
25 .QB88(RegData[88]), .QB89(RegData[89]), .QB90(RegData[90]), .QB91(RegData[91]), // 26 .Read Data	23DA(InputGPR[8]),DA9(InputGPR[9]),DA10(InputGPR[1]),DA11(InputGPR[1]),DA10(InputGPR[1]),
27 .QB92(RegData[92]), .QB93(RegData[93]), .QB94(RegData[94]), .QB95(RegData[95]), // 28 .Read Data	26 // Write Data
29 .QB96(RegData[96]), .QB97(RegData[97]), .QB98(RegData[98]), .QB99(RegData[99]), // 30 Read Data	27 .DA12(InputGPR[12]), .DA13(InputGPR[13]), .DA14(InputGPR[14]), 28 .DA15(InputGPR[15]), // Write Data
31 .QB100(RegData[100]), .QB101(RegData[101]), .QB102(RegData[102]), 32 .QB103(RegData[103]), // Read Data	29 .DA16(InputGPR[16]), .DA17(InputGPR[17]), .DA18(InputGPR[18]), 30 .DA19(InputGPR[19]), // Write Data
32 - QSTOS(registing ros)), # recta Sum	31DA20(InputGPR[20]),DA21(InputGPR[21]),DA22(InputGPR[22]), 32DA23(InputGPR[23]), // Write Data
Page 9 of 16 Ex. 2105 - macc_gpr.v	Page 10 of 16 Ex. 2105 - mace_gpr.v
1 .DA24(InputGPR[24]), .DA25(InputGPR[25]), .DA26(InputGPR[26]), 2 .DA27(InputGPR[27]), // Write Data	1 .DA88(InputGPR[88]), .DA89(InputGPR[89]), .DA90(InputGPR[90]), 2 .DA91(InputGPR[91]), // Write Data
 DA27(InputGPR[27]), // Write Data DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]),
 DA27(InputGPR[27]), // Write Data JDA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), DA31(InputGPR[31]), // Write Data JDA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]),
2 .DA27(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]),	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 6 .DA99(InputGPR[99]), // Write Data 7 .DA100(InputGPR[100]), .DA101(InputGPR[101]), .DA102(InputGPR[102]),
2 .DA27(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[40]), .DA41(InputGPR[41]), .DA42(InputGPR[42]),	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 6 .DA99(InputGPR[90]), // Write Data 7 .DA100(InputGPR[100]), .DA101(InputGPR[101]), .DA102(InputGPR[102]), 8 .DA103(InputGPR[103]), // Write Data 9 .DA104(InputGPR[104]), .DA105(InputGPR[105]), .DA106(InputGPR[106]),
2 .DA27(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[40]), .DA41(InputGPR[41]), .DA42(InputGPR[42]), 10 .DA43(InputGPR[43]), // Write Data 11 .DA44(InputGPR[44]), .DA45(InputGPR[45]), .DA46(InputGPR[46]),	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 6 .DA99(InputGPR[106]), .DA101(InputGPR[101]), .DA102(InputGPR[102]), 7 .DA100(InputGPR[100]), .DA101(InputGPR[101]), .DA102(InputGPR[102]), 8 .DA103(InputGPR[103]), // Write Data 9 .DA104(InputGPR[104]), .DA105(InputGPR[105]), .DA106(InputGPR[106]), 10 .DA107(InputGPR[107]), // Write Data 11 .DA108(InputGPR[108]), .DA109(InputGPR[109]), .DA1101(InputGPR[110]),
2 .DA27(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), DA29(InputGPR[29]), DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), DA33(InputGPR[33]), DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), DA37(InputGPR[37]), DA38(InputGPR[38]), 8 .DA39(InputGPR[40]), DA41(InputGPR[41]), DA42(InputGPR[42]), 10 .DA40(InputGPR[40]), DA41(InputGPR[41]), DA42(InputGPR[42]), 11 .DA44(InputGPR[41]), // Write Data 13 .DA48(InputGPR[48]), DA49(InputGPR[49]), DA50(InputGPR[50]),	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 6 .DA99(InputGPR[96]), .DA101(InputGPR[101]), .DA102(InputGPR[102]), 8 .DA103(InputGPR[103]), // Write Data 9 .DA104(InputGPR[104]), .DA105(InputGPR[105]), .DA106(InputGPR[106]), 10 .DA107(InputGPR[107]), // Write Data 11 .DA108(InputGPR[108]), .DA109(InputGPR[109]), .DA110(InputGPR[110]), 12 .DA111(InputGPR[111]), // Write Data 13 .DA112(InputGPR[111]), .DA113(InputGPR[113]), .DA114(InputGPR[114]),
2 .DA27(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[49]), .DA41(InputGPR[41]), .DA42(InputGPR[42]), 10 .DA43(InputGPR[43]), // Write Data 11 .DA44(InputGPR[44]), .DA45(InputGPR[45]), .DA46(InputGPR[46]), 12 .DA47(InputGPR[47]), // Write Data 13 .DA48(InputGPR[48]), .DA49(InputGPR[49]), .DA50(InputGPR[50]), 14 .DA51(InputGPR[51]), // Write Data 15 .DA52(InputGPR[52]), .DA53(InputGPR[53]), .DA54(InputGPR[54]),	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 6 .DA99(InputGPR[99]), // Write Data 7 .DA100(InputGPR[100]), .DA101(InputGPR[101]), .DA102(InputGPR[102]), 8 .DA104(InputGPR[103]), // Write Data 9 .DA104(InputGPR[103]), // Write Data 10 .DA107(InputGPR[107]), // Write Data 11 .DA108(InputGPR[108]), .DA109(InputGPR[109]), .DA110(InputGPR[101]), 12 .DA111(InputGPR[111]), // Write Data 13 .DA112(InputGPR[112]), .DA113(InputGPR[113]), .DA114(InputGPR[114]), 14 .DA115(InputGPR[115]), // Write Data 15 .DA116(InputGPR[116]), .DA117(InputGPR[117]), .DA118(InputGPR[118]),
2 .DA22(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), DA29(InputGPR[29]), DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), DA33(InputGPR[33]), DA34(InputGPR[34]), 6 .DA35(InputGPR[36]), DA37(InputGPR[37]), DA38(InputGPR[38]), 7 .DA36(InputGPR[36]), DA37(InputGPR[37]), DA38(InputGPR[38]), 8 .DA39(InputGPR[40]), DA41(InputGPR[41]), DA42(InputGPR[42]), 10 .DA40(InputGPR[40]), DA41(InputGPR[41]), DA42(InputGPR[42]), 11 .DA44(InputGPR[44]), DA45(InputGPR[45]), DA46(InputGPR[46]), 12 .DA47(InputGPR[47]), // Write Data 13 .DA48(InputGPR[48]), DA49(InputGPR[49]), DA50(InputGPR[50]), 14 .DA51(InputGPR[51]), // Write Data 15 .DA52(InputGPR[52]), DA53(InputGPR[53]), DA54(InputGPR[54]), 16 .DA55(InputGPR[59]), // Write Data 17 .DA56(InputGPR[59]), // Write Data 19 .DA50(InputGPR[60]), DA57(InputGPR[57]), DA58(InputGPR[58]), 19 .DA50(InputGPR[60]), DA61(InputGPR[61]), DA62(InputGPR[62]),	2 .DA91(InputGPR[91]), // Write Data 3 .DA92(InputGPR[92]), .DA93(InputGPR[93]), .DA94(InputGPR[94]), 4 .DA95(InputGPR[95]), // Write Data 5 .DA96(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 6 .DA99(InputGPR[96]), .DA97(InputGPR[97]), .DA98(InputGPR[98]), 7 .DA100(InputGPR[100]), .DA101(InputGPR[101]), .DA102(InputGPR[102]), 8 .DA103(InputGPR[103]), // Write Data 9 .DA104(InputGPR[104]), .DA105(InputGPR[105]), .DA106(InputGPR[106]), 10 .DA107(InputGPR[107]), // Write Data 11 .DA108(InputGPR[107]), // Write Data 12 .DA111(InputGPR[111]), // Write Data 13 .DA112(InputGPR[112]), .DA113(InputGPR[113]), .DA114(InputGPR[114]), 14 .DA115(InputGPR[115]), // Write Data 15 .DA116(InputGPR[115]), // Write Data 16 .DA119(InputGPR[116]), .DA117(InputGPR[117]), .DA118(InputGPR[118]), 17 .DA120(InputGPR[120]), .DA121(InputGPR[121]), .DA122(InputGPR[122]), 18 .DA124(InputGPR[120]), .DA125(InputGPR[121]), .DA126(InputGPR[122]), 19 .DA124(InputGPR[124]), .DA125(InputGPR[125]), .DA126(InputGPR[126]),
2 .DA28(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA36(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), .DA39(InputGPR[38]), .DA39(InputGPR[38]), .DA39(InputGPR[38]), .DA39(InputGPR[38]), .DA34(InputGPR[38]), .DA43(InputGPR[38]), .DA43(InputGPR[41]), .DA43(InputGPR[41]), .DA43(InputGPR[41]), .DA44(InputGPR[41]), .DA44(InputGPR[41]), .DA45(InputGPR[45]), .DA46(InputGPR[46]), .DA47(InputGPR[47]), .DA49(InputGPR[49]), .DA50(InputGPR[51]), .DA51(InputGPR[51]), .DA51(InputGPR[51]), .DA51(InputGPR[51]), .DA51(InputGPR[51]), .DA53(InputGPR[51]), .DA53(InputGPR[51]), .DA53(InputGPR[51]), .DA54(InputGPR[51]), .DA54(InputGPR[51]), .DA56(InputGPR[56]), .DA57(InputGPR[57]), .DA56(InputGPR[58]), .DA59(InputGPR[58]), .DA59(InputGPR[58]), .DA59(InputGPR[58]), .DA50(InputGPR[58]), .DA50(InputGPR[58]), .DA50(InputGPR[58]), .DA60(InputGPR[61]), .DA60(InputGPR[61]), .DA60(InputGPR[61]), .DA60(InputGPR[61]), .DA60(InputGPR[61]), .DA60(InputGPR[61]), .DA60(InputGPR[64]), .DA60(InputGPR[66]), .DA60(InputGPR[66]), .DA60(InputGPR[64]), .DA60(InputGPR[66]), .DA60(InputGPR[66]	2
2 .DA28(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]),	2
2 .DA28(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA36(InputGPR[35]), ./ Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[49]), .DA41(InputGPR[41]), .DA42(InputGPR[42]), 10 .DA43(InputGPR[43]), // Write Data 11 .DA44(InputGPR[44]), .DA45(InputGPR[45]), .DA46(InputGPR[46]), 12 .DA47(InputGPR[47]), // Write Data 13 .DA48(InputGPR[47]), // Write Data 14 .DA51(InputGPR[51]), // Write Data 15 .DA52(InputGPR[51]), // Write Data 16 .DA52(InputGPR[51]), // Write Data 17 .DA55(InputGPR[52]), .DA53(InputGPR[53]), .DA54(InputGPR[54]), 18 .DA56(InputGPR[56]), .DA57(InputGPR[57]), .DA58(InputGPR[58]), 19 .DA60(InputGPR[60]), .DA61(InputGPR[61]), .DA62(InputGPR[62]), 20 .DA63(InputGPR[61]), // Write Data 21 .DA64(InputGPR[64]), .DA65(InputGPR[65]), .DA66(InputGPR[66]), 22 .DA64(InputGPR[61]), // Write Data 23 .DA68(InputGPR[61]), // Write Data 24 .DA72(InputGPR[67]), // Write Data 25 .DA68(InputGPR[72]), .DA63(InputGPR[73]), .DA74(InputGPR[71]),	2
2 .DA28(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[40]), .DA41(InputGPR[41]), .DA42(InputGPR[42]), 10 .DA43(InputGPR[44]), .DA43(InputGPR[43]), // Write Data 11 .DA44(InputGPR[44]), .DA45(InputGPR[45]), .DA46(InputGPR[46]), 12 .DA47(InputGPR[48]), .DA49(InputGPR[49]), .DA50(InputGPR[50]), 13 .DA48(InputGPR[48]), .DA49(InputGPR[49]), .DA50(InputGPR[50]), 14 .DA51(InputGPR[51]), // Write Data 15 .DA52(InputGPR[51]), // Write Data 16 .DA56(InputGPR[52]), .DA57(InputGPR[57]), .DA58(InputGPR[54]), 17 .DA56(InputGPR[56]), .DA57(InputGPR[67]), .DA58(InputGPR[58]), 18 .DA59(InputGPR[69]), .DA67(InputGPR[61]), .DA62(InputGPR[63]), // Write Data 19 .DA64(InputGPR[63]), // Write Data 19 .DA64(InputGPR[63]), .DA65(InputGPR[61]), .DA62(InputGPR[62]), 10 .DA63(InputGPR[63]), .DA64(InputGPR[64]), .DA66(InputGPR[67]), .DA66(InputGPR[67]), .DA67(InputGPR[67]), .DA67(InputGPR[71]), .DA71(InputGPR[71]), .DA71(InputGPR[71]), .DA71(InputGPR[71]), .DA71(InputGPR[73]), .DA78(InputGPR[73]),	2 DA91(InputGPR[91]), // Write Data 3 DA92(InputGPR[92]), DA93(InputGPR[93]), DA94(InputGPR[94]), 4 DA95(InputGPR[95]), // Write Data 5 DA96(InputGPR[96]), DA97(InputGPR[97]), DA98(InputGPR[98]), 6 DA99(InputGPR[96]), DA97(InputGPR[97]), DA98(InputGPR[98]), 7 DA100(InputGPR[100]), DA101(InputGPR[101]), DA102(InputGPR[102]), 8 DA103(InputGPR[103]), // Write Data 9 DA104(InputGPR[107]), // Write Data 11 DA108(InputGPR[107]), // Write Data 11 DA108(InputGPR[108]), DA109(InputGPR[109]), DA110(InputGPR[106]), 12 DA111(InputGPR[112]), DA113(InputGPR[113]), DA114(InputGPR[114]), 14 DA112(InputGPR[112]), DA113(InputGPR[113]), DA114(InputGPR[114]), 15 DA116(InputGPR[116]), // Write Data 16 DA119(InputGPR[116]), DA117(InputGPR[117]), DA118(InputGPR[118]), 16 DA119(InputGPR[119]), // Write Data 17 DA120(InputGPR[120]), DA121(InputGPR[121]), DA122(InputGPR[122]), 18 DA123(InputGPR[123]), // Write Data 19 DA124(InputGPR[124]), DA125(InputGPR[125]), DA126(InputGPR[126]), 20 DA127(InputGPR[127]), // Write Data 21 // WRITE TEST SIGNALS 22 BISTEA(vss), 23 .TWEA(vss),
2 .DA28(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), // Write Data 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA35(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), .DA38(InputGPR[38]), // Write Data 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[49]), .DA41(InputGPR[41]), .DA42(InputGPR[42]), .DA43(InputGPR[43]), // Write Data 11 .DA44(InputGPR[44]), // Write Data 12 .DA43(InputGPR[47]), // Write Data 13 .DA48(InputGPR[48]), .DA49(InputGPR[49]), .DA50(InputGPR[50]), .DA51(InputGPR[51]), // Write Data 15 .DA52(InputGPR[52]), .DA53(InputGPR[53]), .DA54(InputGPR[54]), .DA54(InputGPR[54]), .DA56(InputGPR[59]), // Write Data 16 .DA56(InputGPR[59]), // Write Data 17 .DA56(InputGPR[59]), // Write Data 18 .DA66(InputGPR[69]), .DA57(InputGPR[57]), .DA58(InputGPR[59]), .DA59(InputGPR[69]), .DA66(InputGPR[69]), .DA66(InputGPR[69]), .DA63(InputGPR[69]), .DA64(InputGPR[61]), .DA64(InputGPR[61]), .DA64(InputGPR[61]), .DA64(InputGPR[61]), .DA67(InputGPR[61]), .DA67(InputGPR[61]), .DA67(InputGPR[61]), .DA67(InputGPR[61]), .DA67(InputGPR[61]), .DA69(InputGPR[61]), .DA68(InputGPR[61]),	2
2 .DA28(InputGPR[27]), // Write Data 3 .DA28(InputGPR[28]), .DA29(InputGPR[29]), .DA30(InputGPR[30]), 4 .DA31(InputGPR[31]), // Write Data 5 .DA32(InputGPR[32]), .DA33(InputGPR[33]), .DA34(InputGPR[34]), 6 .DA36(InputGPR[35]), // Write Data 7 .DA36(InputGPR[36]), .DA37(InputGPR[37]), .DA38(InputGPR[38]), 8 .DA39(InputGPR[39]), // Write Data 9 .DA40(InputGPR[40]), .DA41(InputGPR[41]), .DA42(InputGPR[42]), 10 .DA43(InputGPR[44]), .DA43(InputGPR[43]), .DA44(InputGPR[44]), .DA44(InputGPR[44]), .DA44(InputGPR[44]), .DA47(InputGPR[49]), .DA46(InputGPR[46]), 11 .DA48(InputGPR[48]), .DA49(InputGPR[49]), .DA50(InputGPR[50]), .DA51(InputGPR[51]), // Write Data 13 .DA48(InputGPR[51]), // Write Data 14 .DA51(InputGPR[51]), // Write Data 15 .DA52(InputGPR[52]), .DA53(InputGPR[53]), .DA54(InputGPR[54]), .DA54(InputGPR[51]), .DA59(InputGPR[56]), .DA57(InputGPR[67]), .DA58(InputGPR[68]), .DA59(InputGPR[69]), .DA59(InputGPR[69]), .DA63(InputGPR[63]), // Write Data 16 .DA64(InputGPR[64]), .DA65(InputGPR[65]), .DA66(InputGPR[66]), .DA66(InputGPR[67]), // Write Data 17 .DA64(InputGPR[64]), .DA65(InputGPR[65]), .DA66(InputGPR[66]), .DA66(InputGPR[67]), // Write Data 18 .DA64(InputGPR[67]), // Write Data 19 .DA64(InputGPR[64]), .DA65(InputGPR[65]), .DA66(InputGPR[67]), .DA67(InputGPR[67]), .DA67(InputGPR[67]), .DA67(InputGPR[67]), .DA67(InputGPR[67]), .DA64(InputGPR[71]), .	2 DA91(InputGPR[91]), // Write Data 3 DA92(InputGPR[92]), DA93(InputGPR[93]), DA94(InputGPR[94]), 4 DA95(InputGPR[95]), // Write Data 5 DA96(InputGPR[96]), DA97(InputGPR[97]), DA98(InputGPR[98]), 6 DA99(InputGPR[96]), DA97(InputGPR[97]), DA98(InputGPR[98]), 7 DA100(InputGPR[100]), DA101(InputGPR[101]), DA102(InputGPR[102]), 8 DA103(InputGPR[103]), // Write Data 9 DA104(InputGPR[107]), // Write Data 11 DA108(InputGPR[108]), DA109(InputGPR[105]), DA110(InputGPR[106]), 12 DA111(InputGPR[108]), DA109(InputGPR[109]), DA110(InputGPR[110]), 13 DA112(InputGPR[112]), DA113(InputGPR[113]), DA114(InputGPR[114]), 14 DA115(InputGPR[112]), // Write Data 15 DA116(InputGPR[116]), // Write Data 16 DA119(InputGPR[116]), // Write Data 17 DA120(InputGPR[116]), DA117(InputGPR[117]), DA118(InputGPR[118]), 18 DA123(InputGPR[120]), DA121(InputGPR[121]), DA122(InputGPR[122]), 19 DA124(InputGPR[123]), // Write Data 10 DA124(InputGPR[127]), // Write Data 11 DA124(InputGPR[127]), // Write Data 12 DA124(InputGPR[127]), // Write Data 13 DA125(InputGPR[127]), // Write Data 14 DA126(InputGPR[127]), // Write Data 15 DA126(InputGPR[127]), // Write Data 16 DA126(InputGPR[127]), // Write Data 17 DA126(InputGPR[127]), // Write Data 18 DA127(InputGPR[127]), // Write Data 19 DA124(InputGPR[127]), // Write Data 19 DA124(InputGPR[127]), // Write Data 10 DA126(InputGPR[126]), DA126(InputGPR[126]), DA126(InputGPR[126]), DA126(InputGPR[126]), DA126(InputGPR[126]), DA126(InputGPR[126]), DA126(InputGPR[126]), TADRA3(gpr_wr_addr[0]), TADRA3(gpr_wr_addr[1]), TADRA4(gpr_wr_addr[1]), TADRA6(gpr_wr_addr[1]), TADRA6(gpr_wr_addr[6]), // Write Data
DA28(InputGPR[27]), // Write Data DA28(InputGPR[28]), // Write Data DA32(InputGPR[31]), // Write Data DA32(InputGPR[31]), // Write Data DA33(InputGPR[32]), // Write Data DA33(InputGPR[32]), // Write Data DA33(InputGPR[33]), // Write Data DA34(InputGPR[34]), // Write Data DA36(InputGPR[39]), // Write Data DA41(InputGPR[47]), // Write Data DA44(InputGPR[49]), // Write Data DA44(InputGPR[44]), // Write Data DA44(InputGPR[44]), // Write Data DA44(InputGPR[47]), // Write Data DA44(InputGPR[47]), // Write Data DA44(InputGPR[47]), // Write Data DA45(InputGPR[49]), // Write Data DA55(InputGPR[51]), // Write Data DA55(InputGPR[51]), // Write Data DA56(InputGPR[51]), // Write Data DA56(InputGPR[59]), // Write Data DA56(InputGPR[59]), // Write Data DA66(InputGPR[69]), // Write Data DA66(InputGPR[69]), // Write Data DA66(InputGPR[61]), // Write Data DA66(InputGPR[71]), // Write Data DA66(InputGPR[71	2

	TDA9/I-mu/CDD(9)	TDA10/Janus/CRR(10)		TDA 79/Lange/CDB/79/N TD 199/7 CDD/77/N	TDA74/law (CDD(74))
1 2	.TDA8(InputGPR[9]), .TDA9(InputGPR[9]), .TDA11(InputGPR[11]), // Write Test Data	.TDA10(InputGPR[10]),	1 2	.TDA72(InputGPR[72]), .TDA73(InputGPR[73]), .TDA75(InputGPR[75]), // Write Test Data	.TDA74(InputGPR[74]),
3 4	.TDA12(InputGPR[12]), .TDA13(InputGPR[13]), .TDA15(InputGPR[15]), // Write Test Data	.TDA14(InputGPR[14]),	3 4	.TDA76(InputGPR[76]), .TDA77(InputGPR[77]), .TDA79(InputGPR[79]), // Write Test Data	.TDA78(InputGPR[78]),
5 6	.TDA16(InputGPR[16]), .TDA17(InputGPR[17]), .TDA19(InputGPR[19]), // Write Test Data	.TDA18(InputGPR[18]),	5 6	.TDA80(InputGPR[80]), .TDA81(InputGPR[81]), .TDA83(InputGPR[83]), // Write Test Data	.TDA82(InputGPR[82]),
7 8	.TDA20(InputGPR[20]), .TDA21(InputGPR[21]), .TDA23(InputGPR[23]), // Write Test Data	.TDA22(InputGPR[22]),	7 8	.TDA84(InputGPR[84]), .TDA85(InputGPR[85]), .TDA87(InputGPR[87]), // Write Test Data	.TDA86(InputGPR[86]),
9 10	.TDA24(InputGPR[24]), .TDA25(InputGPR[25]), .TDA27(InputGPR[27]), // Write Test Data	.TDA26(InputGPR[26]),	9 10	.TDA88(InputGPR[88]), .TDA89(InputGPR[89]), .TDA91(InputGPR[91]), // Write Test Data	.TDA90(InputGPR[90]),
11 12	.TDA28(InputGPR[28]), .TDA29(InputGPR[29]), .TDA31(InputGPR[31]), // Write Test Data	.TDA30(InputGPR[30]),	11 12	.TDA92(InputGPR[92]), .TDA93(InputGPR[93]), .TDA95(InputGPR[95]), // Write Test Data	.TDA94(InputGPR[94]),
13 14	.TDA32(InputGPR[32]), .TDA33(InputGPR[33]), .TDA35(InputGPR[35]), // Write Test Data	.TDA34(InputGPR[34]),	13 14	.TDA96(InputGPR[96]), .TDA97(InputGPR[97]), .TDA99(InputGPR[99]), // Write Test Data	.TDA98(InputGPR[98]),
15 16	.TDA36(InputGPR[36]), .TDA37(InputGPR[37]), .TDA39(InputGPR[39]), // Write Test Data	.TDA38(InputGPR[38]),	15 16	.TDA100(InputGPR[100]), .TDA101(InputGPR[101]), .TDA103(InputGPR[103]), // Write Test Data	.TDA102(InputGPR[102]),
17 18	.TDA40(InputGPR[40]), .TDA41(InputGPR[41]), .TDA43(InputGPR[43]), // Write Test Data	.TDA42(InputGPR[42]),	17 18	.TDA104(InputGPR[104]), .TDA105(InputGPR[105]), .TDA107(InputGPR[107]), // Write Test Data	.TDA106(InputGPR[106]),
19 20	.TDA44(InputGPR[44]), .TDA45(InputGPR[45]), .TDA47(InputGPR[47]), // Write Test Data	.TDA46(InputGPR[46]),	19 20	.TDA108(InputGPR[108]), .TDA109(InputGPR[109]), .TDA111(InputGPR[111]), // Write Test Data	.TDA110(InputGPR[110]),
21 22	.TDA48(InputGPR[48]), .TDA49(InputGPR[49]), .TDA51(InputGPR[51]), // Write Test Data	.TDA50(InputGPR[50]),	21 22	.TDA112(InputGPR[112]), .TDA113(InputGPR[113]), .TDA115(InputGPR[115]), // Write Test Data	.TDA114(InputGPR[114]),
23 24	.TDA52(InputGPR[52]), .TDA53(InputGPR[53]), .TDA55(InputGPR[55]), // Write Test Data	.TDA54(InputGPR[54]),	23 24	.TDA116(InputGPR[116]), .TDA117(InputGPR[117]), .TDA119(InputGPR[119]), // Write Test Data	.TDA118(InputGPR[118]),
25 26	.TDA56(InputGPR[56]), .TDA57(InputGPR[57]), .TDA59(InputGPR[59]), // Write Test Data	.TDA58(InputGPR[58]),	25 26	.TDA120(InputGPR[120]), .TDA121(InputGPR[121]), .TDA123(InputGPR[123]), // Write Test Data	.TDA122(InputGPR[122]),
27 28	.TDA60(InputGPR[60]), .TDA61(InputGPR[61]), .TDA63(InputGPR[63]), // Write Test Data	.TDA62(InputGPR[62]),	27 28	.TDA124(InputGPR[124]), .TDA125(InputGPR[125]), .TDA127(InputGPR[127]), // Write Test Data	.TDA126(InputGPR[126]),
29 30	.TDA65(InputGPR[64]), .TDA65(InputGPR[65]), .TDA67(InputGPR[67]), // Write Test Data	.TDA66(InputGPR[66]),	29	//READ TEST SIGNALS	
31 32	.TDA68(InputGPR[68]), .TDA69(InputGPR[69]), .TDA71(InputGPR[71]), // Write Test Data	.TDA70(InputGPR[70]),	30 31	.BISTEB(vss), .TOEB(vss),	
32	. 13.17 (Imputor A(7.1)), // write rest bata		32	.TMEB(vss),	
	Page 13 of 16	Ex. 2105 - macc_gpr.v		Page 14 of 16	Ex. 2105 - macc_gpr.v
1 2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19	.TADRB2(sq_sp_gpr_rd_addr[2]), .TADRB3(sq_sp_gpr_rd_addr[3]	Bl(sq_sp_gpr_rd_addr[1]),), // Read Test Address B5(sq_sp_gpr_rd_addr[5]),	1 2 3 4 5 6 7 8 9		
20 21 22 23 24 25 26	Page 15 of 16	Ex. 2105 - macc_gpr.v		Page 16 of 16	Ex. 2105 - mace_gpr.v

```
1 //
                     -*- Mode: Verilog -*-
                                                                                                                  1 sp1_sx_exp_pvalid, sp0_sx_exp_alu_id, sp1_sx_exp_alu_id,
 2 // Filename : export buffer.v
                                                                                                                  2 sp0 sx exporting, sp1 sx exporting, sp0 sx exp dest,
 3\ \ /\!/\ Description\ \ : The module implements the control for export buffers as well as the quad 4\ \ \  interface between SC and SX blocks
                                                                                                                       sp1_sx_exp_dest, sp0_sx_data0, sp0_sx_data1, sp0_sx_data2,
                                                                                                                       sp0\_sx\_data3, sp1\_sx\_data0, sp1\_sx\_data1, sp1\_sx\_data2,
 5 // Author : Andi Skende
                                                                                                                  5 sp1_sx_data3, sq_sx_exp_type, sq_sx_exp_number, sq_sx_exp_pix,
 6 // Created On : Thu Apr 4 19:13:50 2002
                                                                                                                  6 sq_sx_exp_state, sq_sx_exp_id, sq_sx_exp_valid, sq_sx_free_done,
                                                                                                                  7 sq_sx_free_id, rb0_sx_quad_rtr, rb1_sx_quad_rtr, rb2_sx_quad_rtr,
 8 // Last Modified On:
                                                                                                                  8 rb3_sx_quad_rtr, rb0_sx_color_rtr, rb1_sx_color_rtr,
 9 // Update Count : 0
                                                                                                                        rb2_sx_color_rtr, rb3_sx_color_rtr, rb0_sx_index, rb1_sx_index,
10 // Status : Unknown, Use with caution!
                                                                                                                 10 rb2_sx_index, rb3_sx_index, rb0_sx_index_send, rb1_sx_index_send,
11
                                                                                                                 11 rb2 sx index send, rb3 sx index send, rb0 sx index op,
12 'timescale 1ns / 1ps
                                                                                                                 12 rb1_sx_index_op, rb2_sx_index_op, rb3_sx_index_op,
13 module export_control(/*AUTOARG*/
                                                                                                                 13 rbbm_sx_soft_reset, rbbm_we, rbbm_wd, rbbm_a, rbbm_be, rbbm_re,
                                                                                                                 14 rbb_rs_in, rbb_rd_in, pa_sx_req, pa_sx_sp_id, pa_sx_offset,
15 sx_sc_quad_rtr, sx_sq_exp_count_rdy, sx_sq_exp_pos_avail,
                                                                                                                 15
                                                                                                                        pa_sx_aux, pa_sx_last
      sx_sq_exp_buf_avail, sx_rb_quad_x, sx_rb_quad_y, sx_rb_quad_mask,
                                                                                                                 16
       sx\_rb\_quad\_type, sx\_rb\_quad\_pixel, sx\_rb\_quad\_index,
                                                                                                                 17
18
      sx_rb0_quad_send, sx_rb1_quad_send, sx_rb2_quad_send,
                                                                                                                 18
19 sx rb3 quad send, sx rb0 color data, sx rb1 color data,
                                                                                                                                           sc_sx_quad_x;
20 sx_rb2_color_data, sx_rb3_color_data, sx_rb0_color_send,
                                                                                                                 20 input [1:0]
                                                                                                                                           sc_sx_quad_y;
21 sx_rb1_color_send, sx_rb2_color_send, sx_rb3_color_send,
                                                                                                                       input [31:0] sc_sx_quad_mask;
                                                                                                                 21
22 sx rb0 index rtr, sx rb1 index rtr, sx rb2 index rtr,
                                                                                                                 22
                                                                                                                        input [1:0]
                                                                                                                                           sc_sx_quad_tilex;
      sx_rb3_index_rtr, sx_pa_send, sx_pa_data,
                                                                                                                 23
                                                                                                                                           sc_sx_quad_tiley;
24
                                                                                                                 24
                                                                                                                       input
                                                                                                                                           sc sx quad send;
25 sc_sx_quad_x, sc_sx_quad_y, sc_sx_quad_mask, sc_sx_quad_tilex,
                                                                                                                                           sx_sc_quad_rtr;
                                                                                                                       output
26 sc_sx_quad_tiley, sc_sx_quad_send, sclk, srst, sp0_sx_exp_pvalid,
                                        Page 1 of 100
                                                                                                                                                           Page 2 of 100
                                                                Ex. 2106 - export control.v
                                                                                                                                                                                 Ex. 2106 - export control.v
                          sclk,srst;
                                                                                                                       input [0:0]
                                                                                                                                           sq_sx_exp_valid; //valid cycle
                                                                                                                       input [0:0]
                                                                                                                                           sq sx free done;
                                                                                                                        input [0:0]
                                                                                                                                           sq_sx_free_id;
      //ANDI ???? revisit this interface
      input [3:0]
                     sp0 sx exp pvalid, sp1 sx exp pvalid; //pixel valid mask
     input sp0_sx_exp_alu_id, sp1_sx_exp_alu_id; //isn't one of these signals redundand ??? ANDI
                                                                                                                        //Export Buffer status control interface
       input [0:0] sp0_sx_exporting, sp1_sx_exporting; //isn't one of these signals edundand ??? ANDI
                                                                                                                  8
                                                                                                                        output
                                                                                                                                          sx sq exp count rdy;
10
      input [5:0]
                        sp0_sx_exp_dest, sp1_sx_exp_dest;
11
                                                                                                                                          sx_sq_exp_pos_avail;
12
                                                                                                                 11
                                                                                                                        output [6:0] sx_sq_exp_buf_avail;
                                                                                                                 12
      input [127:0] sp0_sx_data0,sp0_sx_data1,sp0_sx_data2,sp0_sx_data3;
13
       input [127:0]
                       sp1_sx_data0,sp1_sx_data1,sp1_sx_data2,sp1_sx_data3;
                                                                                                                 13
15
                                                                                                                 14
                                                                                                                        //SX-RB interface
                                                                                                                 15
16
17
      //SQ to SX Control Bus
18
                                                                                                                 17
                                                                                                                       //--Quad Interface----//
                                                                                                                 18
19
       input [1:0]
                       sq_sx_exp_type;
20
      input [1:0]
                         sq_sx_exp_number;
      input [0:0]
                                                                                                                 20 output [1:0] sx_rb_quad_x;
                         sq_sx_exp_pix;
                                                                                                                 21 output [1:0] sx_rb_quad_y;
22
      input [2:0]
                         sq_sx_exp_state; //ANDI ....just a hack
                                                                                                                        output [31:0] sx_rb_quad_mask;
23
                                                                                                                        output [0:0] sx_rb_quad_type;
24
      assign
                          sq_sx_exp_state = 3'b0;
25
                                                                                                                        output [3:0] sx rb quad pixel;
                                                                                                                       output [7:0] sx_rb_quad_index;
26 input [0:0]
                          sq_sx_exp_id;
                                         Page 3 of 100
                                                                                                                                                           Page 4 of 100
                                                                                                                                                                                 Ex. 2106 - export control.v
                                                                Ex. 2106 - export control.v
```

ATI 2106 LG v. ATI IPR2015-00326



```
end
                                                                                                                                               state re <= rbbm re;
 2
             else
                                                                                                                                               state\_rs\_in \mathrel{<=} rbb\_rs\_in;
              begin
                                                                                                                                               state rd in <= rbb rd in
               q0\_sp0\_data0 \mathrel{<=} sp0\_sx\_data0;
                q0\_sp0\_data1 \mathrel{<=} sp0\_sx\_data1;
                                                                                                                                        end // always @ (posedge sclk)
                q0\_sp0\_data2 \mathrel{<=} sp0\_sx\_data2;
                q0_sp0_data3 <= sp0_sx_data3;
                q0\_sp1\_data0 \mathrel{<=} sp1\_sx\_data0;
                q0\_sp1\_data1 \mathrel{<=} sp1\_sx\_data1;
10
                q0\_sp1\_data2 \mathrel{<=} sp1\_sx\_data2;
                                                                                                                                       // The fifo below is two-quad wide to allow for double reads on the read side
11
                q0\_sp1\_data3 \mathrel{<=} sp1\_sx\_data3;
                                                                                                                               11
                                                                                                                                      // For this reason, I write the fifo only every other valid cycle from the SC
12
                q1\_sp0\_data0 \mathrel{<=} q0\_sp0\_data0;
                                                                                                                               12
                                                                                                                                       // Quad control/info bus from SC
13
                q1\_sp0\_data1 \mathrel{<=} q0\_sp0\_data1;
                                                                                                                               13
14
                q1_sp0_data2 <= q0_sp0_data2;
                                                                                                                                       //counts two valid quads before it writes into the skid buffer
15
                                                                                                                               15
                q1 sp0 data3 <= q0 sp0 data3;
                                                                                                                                       reg [0:0] quad valid count;
16
                q1 sp1 data0 <= q0 sp1 data0;
                                                                                                                               16
17
                                                                                                                               17
                ql spl datal <= q0 spl datal;
                                                                                                                                       always @(posedge sclk)
                q1\_sp1\_data2 \mathrel{<=} q0\_sp1\_data2;
18
                                                                                                                               18
                                                                                                                                        begin
19
                q1\_sp1\_data3 \le q0\_sp1\_data3;
                                                                                                                               19
                                                                                                                                             if(srst)
20
                q\_sx\_sc\_rtr \mathrel{<=} sx\_sc\_rtr;
                                                                                                                               20
21
                state\_soft\_reset <= rbbm\_sx\_soft\_reset;
                                                                                                                               21
                                                                                                                                               quad valid count <= 2'b0;
22
                state_we <= rbbm_we;
                                                                                                                               22
23
                state wd <= rbbm wd;
                                                                                                                               23
24
                state\_a \le rbbm\_a;
                                                                                                                               24
                state_be <= rbbm_be;
                                                                                                                               25
                                                                                                                                               if(sc_sx_quad_send)
                                               Page 9 of 100
                                                                                                                                                                             Page 10 of 100
                                                                        Ex. 2106 - export_control.v
                                                                                                                                                                                                        Ex. 2106 - export_control.v
                 quad_valid_count <= quad_valid_count+1;
                                                                                                                                               q_quad_x <= sc_sx_quad_x;
2
              end
                                                                                                                                               q_quad_y <= sc_sx_quad_y;
         end // always @ (posedge sclk)
                                                                                                                                               q\_quad\_mask \mathrel{<=} sc\_sx\_quad\_mask;
                                                                                                                                               q\_quad\_tilex <= sc\_sx\_quad\_tilex;
                                                                                                                                               q\_quad\_tiley \mathrel{<=} sc\_sx\_quad\_tiley;
       reg [1:0]
                             q_quad_x;
                                                                                                                                               q\_quad\_send <= sc\_sx\_quad\_send;
        reg [1:0]
                             q_quad_y;
        reg [31:0]
                             q_quad_mask;
                                                                                                                                        end // always @ (posedge sclk)
        reg [1:0]
                             q_quad_tilex;
10
                             q_quad_tiley;
                                                                                                                               10
11
                             q_quad_send;
                                                                                                                               11
                                                                                                                                       wire [38:0] quad_data;
12
                                                                                                                               12
                                                                                                                                       wire [38:0] q_quad_data;
13
       always @(posedge sclk)
                                                                                                                                       wire [77:0] double_quad;
14
         begin
                                                                                                                                       wire [0:0] quad_write;
15
                                                                                                                               15
16
                                                                                                                                      //meaning of each bit going into quad buffer
              begin
17
               q_quad_x <= 2'b0;
                                                                                                                                      //sc_sx_quad1_x 1:0
18
                q_quad_y <= 2'b0;
                                                                                                                                      //sc_sx_quad1_y 3:2
19
                q_quad_mask <= 32'b0;
                                                                                                                                      //sc_sx_quad1_mask 35:4
               q_quad_tilex <= 2'b0;
20
                                                                                                                               20
                                                                                                                                      //sc sx quad1 tilex 37:36
21
               q_quad_tiley <= 1'b0;
                                                                                                                               21 //sc sx quad1 tiley 38
22
                q_quad_send <= 1'b0;
                                                                                                                               22 //sc sx guad0 x 40:39
23
                                                                                                                               23
                                                                                                                                      //sc_sx_quad0_y 42:41
              end
24
                                                                                                                               24
                                                                                                                                      //sc sx quad0 mask 74:43
             else
25
              begin
                                                                                                                               25
                                                                                                                                      //sc_sx_quad0_tilex 76:75
                                              Page 11 of 100
                                                                                                                                                                             Page 12 of 100
                                                                        Ex. 2106 - export control.v
                                                                                                                                                                                                       Ex. 2106 - export control.v
```

```
//sc_sx_quad0_tiley 77
                                                                                                                                                                                    .read rts(read rts), //quad command data available in
                                                                                                                                            the buffer
2
                                                                                                                                                                                    .read_rtr(read_quad_cmd_rtr),
      assign \\ \{sc\_sx\_quad\_tiley,sc\_sx\_quad\_tilex,sc\_sx\_quad\_mask,sc\_sx\_quad\_y,sc\_sx\_quad\_x\};
                                                                                                                                                                                    .read_data(read_data),
        assign \qquad q\_quad\_data = \{q\_quad\_tiley, q\_quad\_tilex, q\_quad\_mask, q\_quad\_y, q\_quad\_x\};
        assign double_quad = {q_quad_data,quad_data};
                                                                                                                                                                                    .clk(sclk),
                                                                                                                                                                                                     //clock and reset
        //second valid quad of the pair coming from SC
        assign quad_write = sc_sx_quad_send & quad_valid_count;
10
11
        assign sx sc quad rtr = q sx sc rtr; //do I need another register stage here ????ANDI
12
                                                                                                                                             //SC-SX interface skid buffer internal read
                                                                                                                                     13
13
                read quad cmd rtr;
        wire [77:0] read data;
                                                                                                                                     14
14
15
                                                                                                                                     15
                                                                                                                                             wire [0:0] valid export;
16
                                                                                                                                     16
                                                                                                                                              wire [0:0] valid pixel export;
17
                                                                                                                                              reg\ [0:0]\quad q0\_valid\_export, q1\_valid\_export\ ; //we\ have\ valid\ exports\ coming\ out\ of\ SPs
                                                                                                                                           \begin{array}{lll} reg & [0:0] & q0\_valid\_pixel\_export, \\ q1\_valid\_pixel\_export, q2\_valid\_pixel\_export, \\ q1\_valid\_pixel\_export, q3\_valid\_pixel\_export, \\ \end{array}
18
        //skid buffer at SC-SX quad control/info data
19
                                                                                                                                     20
                                                                                                                                              reg [0:0] valid_data;
20
        skid_buff_top #(78,128) sc_sx_quad_skid(
                                                                                                                                     21
21
                                              .write_rts(quad_write), //sc is ready to send
                                                                                                                                     22
                                                                                                                                              //if there's quad data in the buffer and valid data coming from SP then pop the fifo.
                                              .write_rtr(sx_sc_rtr), //sx is ready to receive ...fifo not
22
23
                                                                                                                                     23
                                                                                                                                              always @(posedge sclk)
24
                                              .write_data(double_quad), // quad info coming from sc
                                                                                                                                     24
25
                                                                                                                                     25
                                                                                                                                                   if(read_rts & valid_pixel_export)
                                                                                                                                                     begin
                                                Page 13 of 100
                                                                                                                                                                                      Page 14 of 100
                                                                            Ex. 2106 - export_control.v
                                                                                                                                                                                                                  Ex. 2106 - export_control.v
                read_quad_cmd_rtr <= 1'b1;
                                                                                                                                                   if(srst)
 2
                valid data <= q0 valid export;
                                                                                                                                                    begin
                                                                                                                                                      q0_valid_export <= 1'b0;
               end
              else
                                                                                                                                                      q1 valid export <= 1'b0;
              begin
                read quad cmd rtr <= 1'b0;
                                                                                                                                                   else
                 valid data <= 1'b0:
               end
                                                                                                                                                      q0 valid export <= valid export;
         end // always @ (posedge sclk)
                                                                                                                                                      q1\_valid\_export <= q0\_valid\_export;
10
                                                                                                                                     10
      //This valid_export bit is passed through with data into the ALPHA/RGBA test logic or pipeline once depending on ALPHA/RGBA logic latency).
                                                                                                                                     11
                                                                                                                                               end // always @ (posedge sclk)
                                                                                                                                     12
      //The output of it is than send to the read side of the skid buffer sitting at the quad interface between the SX and SC...
                                                                                                                                     13
                                                                                                                                     14
                                                                                                                                             //delay the valid by one cycle to account for ALPHA/RGBA data processing logic latency
15
        //so the respective quad info is read from the quad buffer
                                                                                                                                     15
                                                                                                                                              always @(posedge sclk)
16
                                                                                                                                              begin
        assign valid_export = |sp0_sx_exporting; //exporing pixel or vertices vs. no exports
17
                                                                                                                                                   if(srst)
18
                                                                                                                                     18
19
        //hack for now ...ANDI
                                                                                                                                                      q0_valid_pixel_export <= 1'b0;
20
        //we have an export from shader pipe and this is a pixel export
                                                                                                                                     20
                                                                                                                                                      q1 valid pixel export <= 1'b0;
21
        reg [0:0] q_exp_pix_alu0;
                                                                                                                                                     q2_valid_pixel_export <= 1'b0;
                                                                                                                                     21
22
        assign\ valid\_pixel\_export = valid\_export\ \&\ q\_exp\_pix\_alu0;
                                                                                                                                     22
                                                                                                                                                      q3 valid pixel export <= 1'b0;
23
                                                                                                                                     23
24
        //delay the valid by one cycle to account for ALPHA/RGBA data processing logic latency
                                                                                                                                     24
25
        always @(posedge sclk)
                                                                                                                                                   else
26
         begin
                                                                                                                                     25
                                                                                                                                                    begin
                                                Page 15 of 100
                                                                                                                                                                                      Page 16 of 100
                                                                            Ex. 2106 - export control.v
                                                                                                                                                                                                                 Ex. 2106 - export control.v
```

```
q0 valid pixel export <= valid pixel export;
                                                                                                                                             //four RBs present in the current configuration ...hadwired for the time-being
 2
                 ql\_valid\_pixel\_export <= q0\_valid\_pixel\_export;
                                                                                                                                       2
                                                                                                                                             assign rb sys config = 4'b1111;
                 q2\_valid\_pixel\_export <= q1\_valid\_pixel\_export;
                 q3_valid_pixel_export <= q2_valid_pixel_export;
                                                                                                                                             reg [77:0] quad_data_read ;
         end // always @ (posedge sclk)
                                                                                                                                              //latching the output of the quad command skid buffer before going into the quad command
                                                                                                                                              //with the modified data...modified mask and rb id added
                                                                                                                                              always @(posedge sclk)
        //Next....the quad data just read from the "quad buffer" goes into
                                                                                                                                               begin
        //the "detailed quad buffer" after the 32-bit mask has been modified
                                                                                                                                      11
                                                                                                                                                    if(read quad cmd rtr)
11
       //accordingly based on the result of the ALPHA/RGBA >1.0/0.0 test logic
                                                                                                                                                     quad_data_read <= read_data;
12
       //The tile_x and tile_y informantion as been stripped out of the quad packet.
                                                                                                                                      13
13
        //A two bit id has been attached based on which RB the quad belongs to.
                                                                                                                                      14
        //Also, a index on where the quad resides into Export Buffer has been attached to
                                                                                                                                      15
15
        //the "detailed quad packet" written into the "detailed quad buffer".
                                                                                                                                      16
16
                                                                                                                                      17
                                                                                                                                              reg [6:0] q_export_index;
17
        wire [6:0] export index:
                                                                                                                                      18
                                                                                                                                              always @(posedge sclk)
      wire [1:0] rb_id0,rb_id1; //used to identify which rb the quad belongs to based on the state and math on tile_x and tile_y data
                                                                                                                                      19
20
                                                                                                                                      20
                                                                                                                                                   q_export_index <= export_index;
21
        wire [1:0] tile_x0; //first quad tile coordinates
                                                                                                                                     21
22
        wire [0:0] tile v0;
                                                                                                                                     22
23
        wire [1:0] tile_x1; //second quad tile coordinates
                                                                                                                                     23
24
         wire [0:0] tile y1;
                                                                                                                                      24
                                                                                                                                             //the meaning of each bit going into quad buffer
25
                                                                                                                                      25
                                                                                                                                              //sc_sx_quad1_x 1:0
         wire [3:0] rb_sys_config;
                                                                                                                                              //sc_sx_quad1_y 3:2
                                                Page 17 of 100
                                                                                                                                                                                      Page 18 of 100
                                                                            Ex. 2106 - export_control.v
                                                                                                                                                                                                                  Ex. 2106 - export_control.v
       //sc_sx_quad1_mask 35:4
                                                                                                                                             //delay the read_data bus by one cycle to compensate for the above delay
 2
       //sc sx quad1 tilex 37:36
                                                                                                                                              reg [77:0] q quad data read;
        //sc_sx_quad1_tiley 38
                                                                                                                                              always @(posedge sclk)
        //sc_sx_quad0_x 40:39
                                                                                                                                               begin
        //sc sx quad0 y 42:41
                                                                                                                                                    q_quad_data_read <= quad_data_read;
        //sc sx quad0 mask 74:43
        //sc_sx_quad0_tilex 76:75
        //sc_sx_quad0_tiley 77
                                                                                                                                             //modify the mask based on the ALPHA/RGBA test outcome and pyalid bits.
10
                                                                                                                                      10
11
        assign tile_x1 = quad_data_read[37:36];
                                                                                                                                      11
                                                                                                                                             //remember ....the quad info out of "quad buffer" is in the following format
12
        assign tile_y1 = quad_data_read[38];
                                                                                                                                      12
                                                                                                                                             //{iquad_tiley,iquad_tilex,iquad_mask,iquad_y,iquad_x};
13
                                                                                                                                            //the new format going into "detailed quad buffer" will be {rb_id,export_index,iquad_mask,iquad_y,iquad_x}
14
        assign tile_x0 = quad_data_read[76:75];
                                                                                                                                      15
15
        assign tile_y0 = quad_data_read[77];
                                                                                                                                             wire [91:0] detailed_quad_data;
16
                                                                                                                                      17
                                                                                                                                              wire [7:0] export_index0, export_index1;
17
         //this module is used to define depending on RB/MC config, wich RB unit the current quad
                                                                                                                                      19
                                                                                                                                                         export index0 = {1'b0,q export index[6:0]};
19
      (tile x, tile y) belongs to
                                                                                                                                             assign
                                                                                                                                             assign export_index1 = {1'b1,q_export_index[6:0]};
20
       //there's a one cycle delay through the rb id calculation logic as well as the ALPHA test logic
                                                                                                                                      20
                                        urb id0(.block id(rb id0).
                                                                                                                                      21
21
22
                                                                                       .tile x(tile x0),
       .tile_y(tile_y0),.config(rb_sys_config),.sclk(sclk)); //quad 0
                                                                                                                                      22
                                                                                                                                             //for now there's no need to register the quad_data_read by one cycle
       \begin{array}{cccc} \bmod \underline{rb} \ id & urb \ \underline{id1} (.block \underline{id} (rb \underline{id1}), \\ .tile \underline{y} (tile \underline{y1}),.config (rb \underline{sys} \underline{config}),.sclk (sclk)); //quad \ 1 \end{array}
                                                                                        .tile_x(tile_x1),
                                                                                                                                      23
                                                                                                                                             //the latency through mode_rb_id module is less than one cycle
                                                                                                                                           \begin{array}{lll} assign & detailed\_quad\_data = \{rb\_id0, & export\_index0, quad\_data\_read[74:43], \\ quad\_data\_read[42:41], quad\_data\_read[40:39], \end{array}
25
26
                                                Page 19 of 100
                                                                                                                                                                                      Page 20 of 100
                                                                            Ex. 2106 - export control.v
                                                                                                                                                                                                                  Ex. 2106 - export control.v
```

```
export index1,quad data read[35:4].
                                                                                                                                                                        clk(sclk)
                                                                                                                                                                                        //clock and reset
      quad_data_read[3:2],quad_data_read[1:0]};
                                                                                                                                                                         .reset(srst)
       //writting the detailed quad data into the "detailed quad buffer"
       wire [91:0] quad_data_rb;
                                                                                                                             reg~[0:0] \qquad exp\_data\_in\_buff0, exp\_data\_in\_buff1; \qquad //this~bit~signifies~that~all~the~export~data~for~that~quad~is~in~the~Export~Buffers
       wire [0:0] detailed_quad_valid;
       wire [0:0] read_detailed_quad;
                                                                                                                              wire [0:0] rb_ready;
                                                                                                                               wire [0:0] sx_rb_quad_send;
10
                                                                                                                              wire [1:0] sx_rb_id;
11
12
                                                                                                                              wire [0:0] quad data ready;
                                                                                                                       12
       //simple fifo carrying quad data (eventually sent to RBs) with the correct mask values (post
                                                                                                                       13
14
                                                                                                                       14
15
16
       //--
                                                                                                                       15
                                                                                                                              //SX-RB "index" interface
       skid_buff_top #(92,96,0) sx_rb_detailed_quad_data(
17
                                                                                                                              //reading the "detailed quad buffer" and sending the data to RBs on the quad broadcast bus.
                                                                                                                       16
     . write\_rts(q1\_valid\_pixe1\_export), \quad //modified \\ quad command data is being transfered over from the quad buffer to
18
19
                                                                                                                       17
20
                                                                                                                                           rb\_ready \ = \ rb0\_sx\_quad\_rtr \ \& \ rb0\_sx\_quad\_rtr \ \& \ rb0\_sx\_quad\_rtr \ \&
                                                //to the index fifo
21
                                                //.write rtr(sx sc rtr),
                                                                                                                       20
                                                 .write_data(detailed_quad_data), // quad info
                                                                                                                       21
      with the modified 32 bit mask
23
                                                                                                                       22
                                                                                                                              //send the index quad data to the RBs only when all the RBs are ready to receive it
24
25
                                                 .read_rts(detailed_quad_valid), //this quad
     data is eventually send out to RBs once all exports are in
                                                                                                                              //and there's outstanding quads sitting in the export buffers, attributes of which have
26
                                                 .read_rtr(read_detailed_quad),
                                                                                                                       24
                                                                                                                              //been completly exported from SP into SX
27
                                                 . read\_data(quad\_data\_rb),
28
                                                                                                                              //assign read_detailed_quad = rb_ready & quad_data_ready ;
                                           Page 21 of 100
                                                                                                                                                                  Page 22 of 100
                                                                    Ex. 2106 - export_control.v
                                                                                                                                                                                          Ex. 2106 - export_control.v
                                                                                                                              wire [31:0] osx_quad_mask;
2
       reg [0:0] rb quad data count;
                                                                                                                              wire [0:0] osx quad type;
 3
       assign read_detailed_quad = rb_ready & rb_quad_data_count;
                                                                                                                              wire [3:0] osx quad pixel;
4
                                                                                                                              wire [7:0] osx_quad_index;
                                                                                                                            wire [0:0] osx_rb0_quad_send, osx_rb3_quad_send;
 5
       always @(posedge sclk)
                                                                                                                                                                      osx_rb1_quad_send,
                                                                                                                                                                                                 osx_rb2_quad_send,
        begin
                                                                                                                              reg [1:0]
                                                                                                                                           q_osx_quad_x;
             rb quad data count <= 1'b0;
                                                                                                                                           q_osx_quad_y;
            else if(detailed_quad_valid)
                                                                                                                                         q_osx_quad_mask
10
             rb\_quad\_data\_count <= rb\_quad\_data\_count + 1;
                                                                                                                              reg [0:0]
                                                                                                                                           q_osx_quad_type;
11
                                                                                                                              reg [3:0]
                                                                                                                                           q_osx_quad_pixel;
12
                                                                                                                              reg [7:0]
                                                                                                                                           q_osx_quad_index;
13
       wire [0:0] rb_quad_word_mux;
                                                                                                                              reg [0:0]
                                                                                                                                           q_osx_rb0_quad_send, q_osx_rb1_quad_send, q_osx_rb2_quad_send,
14
       assign rb_quad_word_mux = rb_quad_data_count;
                                                                                                                       15
                                                                                                                             q osx rb3 quad send;
15
                                                                                                                       16
     16
17
                                                                                                                       17
18
                                                                                                                              assign
                                                                                                                                           osx_quad_x = (rb_quad_word_mux) ? quad_data_rb[1:0] :quad_data_rb[47:46]
                                                        export_index1,q_quad_data_read[35:4],
                                                                                                                       19
                                                                                                                                           osx_quad_y
                                                                                                                                                                  (rb_quad_word_mux) ? quad_data_rb[3:2]
19
      q\_quad\_data\_read[3:2], q\_quad\_data\_read[1:0]\};
                                                                                                                              :quad_data_rb[49:48]
     // 91:0
                                                                                                                                                                   (rb_quad_word_mux) ?
20
                                                                                                                             assign osx_quad_mask :quad_data_rb[81:50];
                                                                                                                                                                                                 quad_data_rb[35:4]
21
                                                                                                                            23
24
22
       assign
                  sx\_rb\_quad\_send = detailed\_quad\_valid \ \& \ rb\_ready;
23
                  sx\_rb\_id = (rb\_quad\_word\_mux) ? quad\_data\_rb[45:44] : quad\_data\_rb[91:90];
                                                                                                                       25
24
                                                                                                                       26
                                                                                                                                           osx rb0 quad send = sx rb quad send & (~sx rb id[0]) & (~sx rb id[1]);
                                                                                                                              assign
25
       wire [1:0] osx_quad_x;
                                                                                                                                           osx_rb1_quad_send = sx_rb_quad_send & (sx_rb_id[0]) & (~sx rb_id[1]);
                                                                                                                       27
                                                                                                                              assign
       wire [1:0] osx_quad_y;
                                                                                                                                           osx_rb2_quad_send = sx_rb_quad_send & (~sx_rb_id[0]) & (sx_rb_id[1]);
                                                                                                                       28
                                                                                                                              assign
                                           Page 23 of 100
                                                                                                                                                                  Page 24 of 100
                                                                    Ex. 2106 - export control.v
                                                                                                                                                                                          Ex. 2106 - export control v
```

```
1
       assign
                    osx_rb3_quad_send = sx_rb_quad_send & (sx_rb_id[0]) & (sx_rb_id[1]);
                                                                                                                                 end // always @ (posedge sclk)
2
                                                                                                                          2
3
       //registering the outputs of SX to RB(s) interface
4
                                                                                                                          4
                                                                                                                                //assigning the output signals for the SX - RB quad interface
       always @(posedge sclk)
                                                                                                                                assign \qquad sx\_rb\_quad\_x = q\_osx\_quad\_x;
                                                                                                                                            sx_rb_quad_y = q_osx_quad_y;
                                                                                                                                assign
                                                                                                                                            sx_rb_quad_mask = q_osx_quad_mask;
                                                                                                                                         sx_rb_quad_type; //pixel exports vs. pass through data ????? ANDI
                                                                                                                                //assign
              q_osx_rb0_quad_send <= 1'b0;
                                                                                                                                         sx_rb_quad_pixel; // valid bits for the pixels ???? ANDI
10
              q_osx_rb1_quad_send <= 1'b0;
                                                                                                                                            sx\_rb\_quad\_index = q\_osx\_quad\_index;
                                                                                                                                assign
11
              q_osx_rb2_quad_send <= 1'b0;
                                                                                                                         11
                                                                                                                                           sx_rb0_quad_send = q_osx_rb0_quad_send;
                                                                                                                                assign
12
              q_osx_rb3_quad_send <= 1'b0;
                                                                                                                         12
                                                                                                                                assign
                                                                                                                                            sx_rbl_quad_send = q_osx_rbl_quad_send;
13
                                                                                                                         13
                                                                                                                                            sx_rb2_quad_send = q_osx_rb2_quad_send;
                                                                                                                                assign
14
                                                                                                                                            sx_rb3_quad_send = q_osx_rb3_quad_send;
                                                                                                                                assign
15
                                                                                                                         15
            else
16
                                                                                                                         16
             begin
17
                                                                                                                         17
               q_osx_quad_x <= osx_quad_x;
18
               q_osx_quad_y <= osx_quad_y;
                                                                                                                         18
19
               q_osx_quad_mask <= osx_quad_mask;
                                                                                                                         19
                                                                                                                                //SQ SX Export Control (allocation/deallocation) Bus Interface ....registering the data
20
               q\_osx\_quad\_index <= osx\_quad\_index;
                                                                                                                         20
21
               q\_osx\_rb0\_quad\_send <= osx\_rb0\_quad\_send \ ;
                                                                                                                        21
22
               q\_osx\_rb1\_quad\_send <= osx\_rb1\_quad\_send \ ;
                                                                                                                         22
                                                                                                                                //register the last request on thread 0
23
               q\_osx\_rb2\_quad\_send <= osx\_rb2\_quad\_send \ ;
                                                                                                                        23
                                                                                                                                reg [1:0] q_exp_type_alu0;
24
               q\_osx\_rb3\_quad\_send <= osx\_rb3\_quad\_send \ ;
                                                                                                                         24
                                                                                                                                reg [1:0] q_exp_number_alu0;
25
             end
                                                                                                                                reg [2:0] q_exp_state_alu0;
                                           Page 25 of 100
                                                                                                                                                                    Page 26 of 100
                                                                     Ex. 2106 - export_control.v
                                                                                                                                                                                              Ex. 2106 - export_control.v
       reg [0:0] q_exp_alu_id_alu0;
                                                                                                                                wire [3:0] attr count;
2
       reg [0:0] q exp pos alu0;
                                                                                                                                reg [2:0] q_exp_attr_count_alu0, q_exp_attr_count_alu1;
4
5
       reg [0:0] q_exp_alu_id;
                                                                                                                               //a different write pointer for each alu thread
                                                                                                                                reg [6:0] write_alu0_ptr,write_alu1_ptr;
      //register the last request on thread 1
       reg [1:0] q_exp_type_alu1;
       reg [1:0] q_exp_number_alu1;
                                                                                                                              //decoding on what kind of type of export request we are getting ....(pixel vs. vertex vs. memory exports)
                                                                                                                         10
       reg [2:0] q_exp_state_alu1;
10
                                                                                                                         11
11
       reg [0:0] q_exp_alu_id_alu1;
                                                                                                                         12
                                                                                                                                wire [0:0] exporting_pixel;
12
       reg \ [0:0] \qquad q\_exp\_pix\_alu1;
                                                                                                                                             exporting_pixel = ~sq_sx_exp_type[1] & (sq_sx_exp_type[0] |
13
       reg [0:0] q_exp_pos_alu1;
                                                                                                                               ~sq sx exp type[0]);
                                                                                                                         15
       reg [0:0] q_exp_valid;
                                                                                                                         16
                                                                                                                                wire [0:0] exporting position;
16
       reg [6:0] write_alu0_base_ptr, write_alu1_base_ptr;
                                                                                                                         17
                                                                                                                                          exporting_position = ~sq_sx_exp_type[0] & sq_sx_exp_type[1];
                                                                                                                                assign
                   alloc_wr_head_ptr,q_alloc_wr_head_ptr ; //points to where the next allocated
                                                                                                                         18
                                                                                                                         19
19
       reg [6:0] alloc wr tail ptr;//points to where the end of free export space is
                                                                                                                         20
                                                                                                                                always @(posedge sclk)
20
                                                                                                                        21
21
                                                                                                                        22
                                                                                                                                     if(srst)
22
       //point to where the next position allocation space will be when the sequencer asks for it
                                                                                                                        23
23
      //this value is relative to location 0x40 (64 first entries in the buffer belong to pixel data)
                                                                                                                        24
                                                                                                                                        q\_exp\_state\_alu0 <= 3"b0;
24
       reg~[3:0] \qquad pos\_wr\_head\_ptr,q\_pos\_wr\_head\_ptr~;
                                                                                                                         25
                                                                                                                                        q_exp_alu_id_alu0 <= 1'b0;
25
       reg [3:0] pos_wr_tail_ptr ; //points to where the end of free export space is
                                                                                                                                        q_exp_type_alu0 <= 2'b0;
26
                                           Page 27 of 100
                                                                                                                                                                    Page 28 of 100
                                                                    Ex. 2106 - export control.v
                                                                                                                                                                                             Ex. 2106 - export control v
```

```
q_exp_number_alu0 <= 2'b0;
                                                                                                                                                      q\_exp\_type\_alu1 <= sq\_sx\_exp\_type;
 2
                q\_exp\_pix\_alu0 <= 1'b0;
                                                                                                                                                      q\_exp\_number\_alu\,l \mathrel{<=} sq\_sx\_exp\_number;
                q\_exp\_pos\_alu0 <= 1'b0;
                                                                                                                                                      q\_exp\_state\_alu1 <= sq\_sx\_exp\_state;
                q\_exp\_state\_alu1 <= 3'b0;
                                                                                                                                                      q\_exp\_alu\_id\_alu1 \mathrel{<=} sq\_sx\_exp\_id;
               q\_exp\_alu\_id\_alu1 <= 1'b0;
                                                                                                                                                      q\_exp\_attr\_count\_alu1 \mathrel{<=} attr\_count;
               q\_exp\_type\_alu1 <= 2"b0;
                q\_exp\_number\_alu1 <= 2"b0;
                                                                                                                                                      //for the positions, the base address is relative to base 0x40 (64 first entries
                q_exp_pix_alu1 <= 1'b0;
                                                                                                                                                      write_alu1_base_ptr <= (exporting_pixel)? alloc_wr_head_ptr : 7'h40 +
                q_exp_pos_alu1 <= 1'b0;
                                                                                                                                  pos wr head ptr;
                                                                                                                              10
                                                                                                                              11
                                                                                                                                                    end
11
                q_exp_alu_id <= 1'b0;
                                                                                                                              12
                                                                                                                                                   else
12
                q\_exp\_valid <= 1"b0;
                                                                                                                             13
                                                                                                                                                    begin
13
               write_alu0_base_ptr <= 7'b0;
                                                                                                                              14
                                                                                                                                                     q_exp_pix_alu0 <= exporting_pixel;
               write_alu1_base_ptr <= 7'b0;
                                                                                                                              15
                                                                                                                                                     q_exp_pos_alu0 <= exporting_position;
15
                q alloc wr head ptr <= 7'b0;
                                                                                                                              16
                                                                                                                                                     q_exp_type_alu0 <= sq_sx_exp_type
               q\_pos\_wr\_head\_ptr <= 7'b0;
16
                                                                                                                              17
                                                                                                                                                     q\_exp\_number\_alu0 <= sq\_sx\_exp\_number;
17
                                                                                                                                                     q_exp_state_alu0 <= sq_sx_exp_state;
              end
                                                                                                                              18
18
             else
                                                                                                                              19
                                                                                                                                                     q\_exp\_alu\_id\_alu0 \mathrel{<=} sq\_sx\_exp\_id;
19
              begin
                                                                                                                              20
                                                                                                                                                      q\_exp\_attr\_count\_alu0 <= attr\_count;
20
               if(sq_sx_exp_valid)
                                                                                                                             21
21
                begin
                                                                                                                                                      //for the positions, the base address is relative to base 0x40 (64 first entries
22
                    if(sq_sx_exp_id)
                                                                                                                                  24
25
23
                      begin
24
                        q\_exp\_pix\_alu1 \ <= exporting\_pixel;
                                                                                                                             26
                                                                                                                                                    end // else: !if(sq\_sx\_exp\_alu\_id)
                        q\_exp\_pos\_alu1 \ \mathop{<=} \ exporting\_position;
                                                                                                                             27
                                             Page 29 of 100
                                                                                                                                                                           Page 30 of 100
                                                                       Ex. 2106 - export_control.v
                                                                                                                                                                                                     Ex. 2106 - export_control.v
                     q_exp_alu_id <= sq_sx_exp_id;
                                                                                                                                     //from the sequencer is assigned to
                 end // if (sq_sx_exp_valid)
                                                                                                                                     wire [2:0] current id state;
                                                                                                                                                   current_id_state = sq_sx_exp_state;
                q_exp_valid <= sq_sx_exp_valid;
                q_alloc_wr_head_ptr <= alloc_wr_head_ptr;
                                                                                                                                     reg [31:0] current_attr_state;
               q_pos_wr_head_ptr <= pos_wr_head_ptr;
                                                                                                                                     always @(/*AUTOSENSE*/current_id_state or state_import_export0
              end // else: !if(srst)
                                                                                                                                             or state_import_export1 or state_import_export2
        end // always @ (posedge sclk)
                                                                                                                                             or state_import_export3 or state_import_export4
10
                                                                                                                             10
                                                                                                                                             or state_import_export5 or state_import_export6
11
                                                                                                                             11
12
                                                                                                                             12
                                                                                                                             13
        //Export buffer space allocation and managment triggered by an allocation request coming
                                                                                                                                           case(current_id_state)
                                                                                                                                            3'h0:current_attr_state =state_import_export0;
15
       //SQ through the SQ-SX export control interface above-----
                                                                                                                                            3'h1:current_attr_state =state_import_export1;
16
                                                                                                                                           3'h2:current_attr_state =state_import_export2;
17
                                                                                                                                           3'h3:current_attr_state =state_import_export3;
18
                                                                                                                                            3'h4:current_attr_state =state_import_export4;
       //there's 64 entries for pixel data and 16 permanently assigned entries for position/auxiliary
19
20
                                                                                                                                           3'h5:current_attr_state =state_import_export5;
      parameter [7:0] BUFFER_SIZE = 8'h40; //Export buffer size ....set via the instantiation of this module from the top module
                                                                                                                             20
                                                                                                                                            3'h6:current_attr_state =state_import_export6;
21
22
                                                                                                                             21
                                                                                                                                            3'h7:current attr state =state import export7;
23
        parameter [3:0] POS_BUFFER_SIZE = 4'hf;
                                                                                                                             22
                                                                                                                                            default : current attr state =state import export0;
24
                                                                                                                             23
                                                                                                                                           endcase // case(current id state)
25
                                                                                                                             24
                                                                                                                                      end // always @ (...
        //this signal is a place holder of the state if that the current thread allocation request
                                             Page 31 of 100
                                                                                                                                                                           Page 32 of 100
                                                                       Ex. 2106 - export control.v
                                                                                                                                                                                                    Ex. 2106 - export control.v
```

```
reg [2:0] attribute_offset, q_attribute_offset;
                                                                                                                                   wire [1:0] export_type;
                                                                                                                                   reg [0:0] position_aux, q_position_aux;
                                                                                                                                  //00:no export
       //Generation of the write index into the Export Buffers
                                                                                                                                   //01:vertex export
       //Decoding the export data type based on the destination pointer coming down from SP
                                                                                                                                 \begin{array}{lll} assign & export \underline{type} = (sp0\_sx\_exp\_alu\_id) & ? & (sp0\_sx\_exporting[0] & & q\_exp\_pix\_alu1,sp0\_sx\_exporting[0] & \sim q\_exp\_pix\_alu1\}; \end{array}
10
                                                                                                                                            {sp0_sx_exporting[0] &
                                                                                                                                                                             q_exp_pix_alu0,sp0_sx_exporting[0]
                                                                                                                           10
11
11
                                                                                                                                 ~q exp pix alu0};
12
      'define COLOR0 6'h00
                                                                                                                           12
                                                                                                                                   always @(/*AUTOSENSE*/`COLOR0 or `COLOR1 or `COLOR2 or `COLOR3
13
      'define COLOR1 6'h01
                                                                                                                           13
14
      'define COLOR2 6'h02
                                                                                                                           14
                                                                                                                                          or 'COLORFOG0 or 'COLORFOG1 or 'COLORFOG2 or 'COLORFOG3
                                                                                                                                          or 'PIXEL EXPORT or 'POSITION or 'SPRITE EDGE
15
      'define COLOR3 6'h03
                                                                                                                           15
16
      'define COLORFOG0 6'h08
                                                                                                                           16
                                                                                                                                          or `VERTEX_EXPORT or `Z_DATA or export_type
17
      'define COLORFOG1 6'h09
                                                                                                                           17
                                                                                                                                          or sp0_sx_exp_dest)
18
      'define COLORFOG2 6'h0a
                                                                                                                           18
19
      'define COLORFOG3 6'h0b
                                                                                                                           19
20
      'define Z_DATA 6'h3f
                                                                                                                           20
                                                                                                                                          `PIXEL_EXPORT:
21
      'define POSITION 6'h3e
                                                                                                                           21
22
      'define SPRITE_EDGE 6'h3f
                                                                                                                           22
                                                                                                                                            position_aux = 1'b0;
23
      //`define NO_EXPORT 2'h0
                                                                                                                                            case(sp0_sx_exp_dest)
24
      'define PIXEL_EXPORT 2'h2
                                                                                                                           24
25
      'define VERTEX_EXPORT 2'h1
                                                                                                                                                'COLOR1:attribute_offset = 3'h1;
                                                                                                                                                'COLOR2:attribute_offset = 3'h2;
                                            Page 33 of 100
                                                                                                                                                                        Page 34 of 100
                                                                      Ex. 2106 - export_control.v
                                                                                                                                                                                                  Ex. 2106 - export_control.v
                     'COLOR3:attribute_offset = 3'h3;
                                                                                                                                    end // always @ (...
                     'COLORFOG0:attribute offset = 3'h0;
                     'COLORFOG1:attribute offset = 3'h1:
                     'COLORFOG2:attribute offset = 3'h2:
                                                                                                                                    //We mantain two write counts and two write pointers ...one for each ALU thread space
                                                                                                                                 allocated in export buffers.
                    'COLORFOG3:attribute_offset = 3'h3;
                                                                                                                                  //Export Buffer write state machine
                    'Z DATA:attribute_offset = 3'h4;
                                                                                                                                   reg [1:0] write_state,next_write_state;
                 endcase // case(sp0_sx_exp_dest)
                                                                                                                                 reg [2:0] alu0_attrib_count, alu1_attrib_count; //counts up to number of attributes for a pixel (2 full quads accross two shader pipes)
               end // case: VERTEX
              'VERTEX EXPORT:
10
                                                                                                                           11
                                                                                                                                 'define NO_EXPORT 2'b00
11
                case(sp0_sx_exp_dest)
                                                                                                                                 'define EXPORT_ALU0 2'b01
12
                    'POSITION:
                                                                                                                                 'define EXPORT_ALU1 2'b10
13
                                                                                                                           14
                       attribute_offset = 3'h0; // + count of the position vectors that have been
14
15
                                                                                                                           15
                                                                                                                                   always @(posedge sclk)
      exported so far
                                                                                                                           16
                                                                                                                                    begin
16
                       position_aux = 1'b0;
                                                                                                                           17
                                                                                                                                        if(srst)
17
                     end
                                                                                                                           18
18
                     `SPRITE_EDGE:
                                                                                                                                           write state <= 'NO EXPORT;
                                                                                                                           19
19
                                                                                                                           20
                                                                                                                                         end
                       attribute_offset = 3'h4; //starting offset is always relative position 4
20
                                                                                                                           21
21
                       position_aux = 1'b1;
                                                                                                                           22
22
                                                                                                                           23
                                                                                                                                           write_state <= next_write_state;
23
                endcase // case(sp0_sx_exp_dest
                                                                                                                           24
24
               end // case: VERTEX
                                                                                                                           25
                                                                                                                                    end // always @ (posedge sclk)
25
              default : attribute offset = 3'h0:
             endcase // case(sp0_sx_exporting)
                                            Page 35 of 100
                                                                                                                                                                        Page 36 of 100
                                                                      Ex. 2106 - export control.v
                                                                                                                                                                                                 Ex. 2106 - export control.v
```

```
'EXPORT ALU1:
2
       reg_no_export_state, export_alu0_state, export_alu1_state;
                                                                                                                                       begin
3
                                                                                                                                        if(~valid export)
4
       always @(/*AUTOSENSE*/`EXPORT_ALU0 or `EXPORT_ALU1 or `NO_EXPORT
                                                                                                                                            next_write_state = 'NO_EXPORT;
              or sp0_sx_exp_alu_id or valid_export or write_state)
                                                                                                                                         else if(valid_export & \simsp0_sx_exp_alu_id)
                                                                                                                                            next_write_state = `EXPORT_ALU0;
            case(write_state)
             'NO_EXPORT:
                                                                                                                                            next_write_state = `EXPORT_ALU1;
10
                                                                                                                                      default : next_write_state = `NO_EXPORT;
                if(valid\_export \ \& \ (\sim sp0\_sx\_exp\_alu\_id))
11
                  next_write_state = `EXPORT_ALU0;
                                                                                                                                     endcase // case(write_state)
12
                else if(valid_export & sp0_sx_exp_alu_id)
                                                                                                                                 end // always @ (...
13
                   next_write_state = `EXPORT_ALU1;
                                                                                                                        13
15
                   next write state = 'NO EXPORT;
                                                                                                                        15
16
                                                                                                                        16
                                                                                                                               //counting the number of attributes per pixel being transfered from SP into SX
17
             `EXPORT ALU0:
                                                                                                                                //when all the attributes are present in Export Bufffers, the RB "detailed quad buffer" is
                                                                                                                        18
18
                                                                                                                        19
                                                                                                                               //and the quad data is sent to the RB it belongs to.
19
               if(~valid export)
                                                                                                                        20
                                                                                                                                reg [6:0] exported_quads_count; //how many outstanding quads to be sent to RB
20
                   next_write_state = `NO_EXPORT;
                                                                                                                                wire exported_quad_inc; //increment the above count
21
                else if(valid_export & sp0_sx_exp_alu_id)
                                                                                                                        22
                                                                                                                                wire exported_quad_dec; //decrement the above count
22
                   next_write_state = `EXPORT_ALU1;
                                                                                                                        23
23
                                                                                                                        24
                                                                                                                                assign quad_inc_dec = {exported_quad_inc,exported_quad_dec};
24
                   next_write_state = `EXPORT_ALU0;
                                                                                                                        25
                                                                                                                                assign exported_quad_dec = read_detailed_quad;
                                           Page 37 of 100
                                                                                                                                                                    Page 38 of 100
                                                                    Ex. 2106 - export_control.v
                                                                                                                                                                                            Ex. 2106 - export_control.v
       always @(posedge sclk)
                                                                                                                                             begin
2
        begin
                                                                                                                                               alu1 attrib count <= 3'b0;
                                                                                                                                               exp\_data\_in\_buff1 <= 1"b1;
            if(srst)
             begin
               alu0_attrib_count <= 3'b0;
                                                                                                                                              end
               alu1_attrib_count <= 3'b0;
                                                                                                                                             else
             end
            else
                                                                                                                                               alu1 attrib count <= alu1 attrib count + 1;
                                                                                                                                               exp_data_in_buff1 <= 1'b0;
               if(export_alu0_state)
10
                                                                                                                        10
11
                                                                                                                        11
                                                                                                                                        end // if (export_alu1_state)
12
                   if(alu0\_attrib\_count>=q\_exp\_attr\_count\_alu0\ )
                                                                                                                        12
                                                                                                                                      end // else: !if(srst)
13
                                                                                                                                 end // always @ (posedge sclk)
                      alu0_attrib_count <= 3'b0;
                      exp_data_in_buff0 <= 1'b1;
                                                                                                                        15
17
                    else
                                                                                                                        17
18
                                                                                                                                //if any of the thread finished a given pixel...2 quads across the 8 export buffers beloning to
                                                                                                                        19
                      alu0_attrib_count <= alu0_attrib_count + 1;
                                                                                                                        20
                                                                                                                                assign exported_quad_inc = exp_data_in_buff0 | exp_data_in_buff1;
                      exp data in buff0 <= 1'b0;
20
                                                                                                                        21
                                                                                                                                assign quad data ready = exported quads count > 7'b0;
21
                     end
                                                                                                                        22
22
                end // if (export alu0 state)
                                                                                                                        23
23
               if(export alul state)
                                                                                                                        24
24
                begin
                                                                                                                        25
                                                                                                                               //exported_quads_count process
                    if(alu1\_attrib\_count>=q\_exp\_attr\_count\_alu1\ )
                                                                                                                               //the count below is indicating the number of quads outstanding in detailed_quad_buffer that
                                           Page 39 of 100
                                                                                                                                                                   Page 40 of 100
                                                                    Ex. 2106 - export control.v
                                                                                                                                                                                            Ex. 2106 - export control v
```

```
//have been completly exported into Export Buffers but not yet issued to RBs because the
                                                                                                                                       always @(posedge sclk)
                                                                                                                                2
       //ready yet
                                                                                                                                            if(srst)
        always @(posedge sclk)
                                                                                                                                             q\_sp0\_exp\_alu\_id <= 1"b0;
         begin
                                                                                                                                              q\_sp0\_exp\_alu\_id \mathrel{<=} sp0\_sx\_exp\_alu\_id;
                exported_quads_count <= 7'b0;
                                                                                                                                       assign export_index = (q_sp0_exp_alu_id) ? write_alu1_ptr:write_alu0_ptr;
                                                                                                                               10
11
              begin
                                                                                                                               11
                                                                                                                                      reg export_buffer_wen,export_buffer_wew;
12
                case(quad inc dec)
                                                                                                                               12
                                                                                                                                      //counts the number of position export transactions comming from the SP
13
                 00:; //do nothing
                                                                                                                               13
                                                                                                                                      //the count would count up to 4 if only positions, and 8 if positions + sprite/edges
14
                 01: exported quads count <= exported quads count - 1;
                                                                                                                                       //64 positions would come first and then sprite/edges (if state indicates their presence)
15
                 10: exported_quads_count <= exported_quads_count + 2;
                                                                                                                               15
16
                 11:://do nothing
                                                                                                                               16
                                                                                                                                      reg [1:0] pos0 exp count; //first position thread
17
                endcase // case(quad_inc_dec
                                                                                                                               17
                                                                                                                                       reg [1:0] pos1 exp count; //second position thread
18
              end // else: !if(srst)
                                                                                                                               18
                                                                                                                                       wire pos0 exported, pos1 exported;
19
                                                                                                                               19
                                                                                                                                       reg [1:0] q pos0 exp count, q pos1 exp count;
20
                                                                                                                               20
                                                                                                                                       wire pos data ready;
21
       //combinational logic based on write_state current state value
                                                                                                                               21
22
                                                                                                                               22
23
       //final index into the export buffer
                                                                                                                               23
                                                                                                                                       //depending on what kind of export we are dealing with (pixel vs. position)
24
       //this values gets sent to export_buffer logic as a write address
                                                                                                                               24
                                                                                                                                       //the write_alu_ptr is calculated differently
25
        reg q_sp0_exp_alu_id;
                                                                                                                                       //in the case of the position data, the exports are stored sequentially in the upper
                                              Page 41 of 100
                                                                                                                                                                             Page 42 of 100
                                                                        Ex. 2106 - export_control.v
                                                                                                                                                                                                       Ex. 2106 - export_control.v
       //16 entries of the export buffer. In the case of pixel exports depending on the order of the
                                                                                                                                       always @(posedge sclk)
 2
                                                                                                                                2
       //exports the storing is not done in sequential order. Within a buffer assigned to a given thread
                                                                                                                                       begin
       //the colors 0-3 are stored first and then the z vector.
                                                                                                                                            if(srst)
                                                                                                                                              vector count0 <= 2'b00:
                                                                                                                                            else \ if((q0\_valid\_pixel\_export \ | \ q1\_valid\_pixel\_export) \ \& \ export\_alu0\_state)
       //mantaining a vector count per attribute per thread.
                                                                                                                                              vector_count0 <= vector_count0 + 1;
       //in other word : for each attribute being exported we count up to 4 vectors of 16 (128) values
                                                                                                                                7
                                                                                                                                        end
                                                                                                                                8
       //this count would go from 0 to 3.
                                                                                                                                9
                                                                                                                                       always @(posedge sclk)
10
                                                                                                                               10
11
        reg [3:0] vector_count0;
                                                                                                                               11
        reg [3:0] vector_count1;
                                                                                                                               12
                                                                                                                                              vector_count1 <= 2'b00;
13
                                                                                                                               13
                                                                                                                                            else if(valid_pixel_export & export_alul_state)
14
                                                                                                                               14
                                                                                                                                              vector_count1 <= vector_count1 + 1;
15
       //forcing the number of attributes per pixel to 1 ...hack ANDI
                                                                                                                               15
16
       //eventually, the number of attributes per pixel will be factored in
17
       //the sequencer allocation request once we go to the new Sequencer allocation scheme
                                                                                                                               17
18
                                                                                                                               18
                                                                                                                                       reg [6:0] write_pos0_ptr, write_color0_ptr;
19
       parameter [3:0] attr exp count = 1'b1;
                                                                                                                                       reg [6:0] write_pos1_ptr, write_color1_ptr;
20
                                                                                                                               20
21
       wire [5:0] vector wr0 offset;
                                                                                                                               21
22
       wire [5:0]
                      vector_wrl_offset;
                                                                                                                               22
23
                                                                                                                                       //registering the pointer values before assigning them in the case statement tied to the write
24
        assign
                      vector_wr0_offset = attr_exp_count * vector_count0;
                                                                                                                               24
25
                      vector_wrl_offset = attr_exp_count * vector_count l;
                                                                                                                               25
26
                                                                                                                                       always @(posedge sclk)
                                              Page 43 of 100
                                                                                                                                                                             Page 44 of 100
                                                                        Ex. 2106 - export control.v
                                                                                                                                                                                                       Ex. 2106 - export control.v
```

```
1
         begin
                                                                                                                                        case(write state)
 2
             if(srst)
                                                                                                                                         'NO EXPORT:
              begin
                                                                                                                                          begin
                write_pos0_ptr <= 7'b0;
                                                                                                                                            //turn off the write enables
                write\_color0\_ptr <= 7'b0;
                                                                                                                                            no_export_state = 1'b1;
                write_pos1_ptr <= 7'b0;
                                                                                                                                            export_alu0_state = 1'b0;
                write_color1_ptr <= 7'b0;
                                                                                                                                            export_alu1_state = 1'b0;
                                                                                                                                            export_buffer_wen = 1'b0;
                                                                                                                                            export_buffer_wew = 1'b0;
10
                                                                                                                           10
11
               write\_pos0\_ptr \mathrel{<=} write\_alu0\_base\_ptr + q\_attribute\_offset + pos0\_exp\_count;
                                                                                                                                         `EXPORT_ALU0:
12
                write\_color0\_ptr <= write\_alu0\_base\_ptr + q\_attribute\_offset + vector\_wr0\_offset;
                                                                                                                                          begin
13
                write_posl_ptr <= write_alu1_base_ptr + q_attribute_offset + pos1_exp_count;
                                                                                                                                            //turn on the write enables
14
               write_colorl_ptr <= write_alul_base_ptr + q_attribute_offset + vector_wrl_offset;
                                                                                                                                            write_alu0_ptr = (q_exp_pix_alu0) ? write_color0_ptr : write_pos0_ptr;
15
                                                                                                                           15
                                                                                                                                            no export state = 1'b0;
              end
                                                                                                                                            export alu0 state = 1'b1:
16
         end // always @ (posedge sclk)
                                                                                                                           16
17
                                                                                                                           17
                                                                                                                                            export alu1 state = 1'b0:
                                                                                                                                            export buffer wen = 1'b1;
18
                                                                                                                           18
19
                                                                                                                           19
                                                                                                                                            export_buffer_wew = 1'b1;
        always @(/*AUTOSENSE*/`EXPORT_ALU0 or `EXPORT_ALU1 or `NO_EXPORT
20
                                                                                                                           20
                                                                                                                                          end
                                                                                                                                         'EXPORT ALU1:
21
               or pos0_exp_count or pos1_exp_count or q_attribute_offset
                                                                                                                           21
22
               or q_exp_pix_alu0 or q_exp_pix_alu1 or vector_wr0_offset
                                                                                                                           22
23
               or vector_wrl_offset or write_alu0_base_ptr
                                                                                                                           23
                                                                                                                                            //turn on the write enables
24
               or write_alu1_base_ptr or write_state)
                                                                                                                           24
                                                                                                                                            write_alu1_ptr = (q_exp_pix_alu1) ? write_color1_ptr: write_pos1_ptr;
25
         begin
                                                                                                                                            no_export_state = 1'b0;
                                             Page 45 of 100
                                                                                                                                                                        Page 46 of 100
                                                                      Ex. 2106 - export_control.v
                                                                                                                                                                                                 Ex. 2106 - export_control.v
                 export_alu0_state = 1'b0;
                 export alu1 state = 1'b1;
                                                                                                                                  reg [7:0] buffer size;
                export_buffer_wen = 1'b1;
                                                                                                                                   reg [3:0] pos buffer size;
                export_buffer_wew = 1'b1;
                                                                                                                                          exported pos inc;
                                                                                                                                   wire exported_pos_dec;
              default
                                                                                                                                   reg [6:0] exported_pos_count;
                                                                                                                                   wire clipp_outstanding_req;
                //turn off the write enables
                                                                                                                                   wire sx_pa_req_rtr;
                no_export_state = 1'b1;
                                                                                                                                   wire [4:0] pa_req_packet;
10
                export_alu0_state = 1'b0;
                                                                                                                           10
                                                                                                                                   wire pa_pos_req;
11
                 export_alu1_state = 1'b0;
                                                                                                                           11
                                                                                                                                   wire [4:0] pa_req_control;
12
                 export_buffer_wen = 1'b0;
                                                                                                                           12
                                                                                                                                   reg [4:0] q_pa_req_control;
13
                 export_buffer_wew = 1'b0;
                                                                                                                           13
14
                                                                                                                           14
15
             endcase // case(write_state)
                                                                                                                                 //deriving the size of the buffer space required for current thread allocation request coming from
16
         end // always @ (...
                                                                                                                           17
                                                                                                                                  //sequencer via SQ SX exp interface.
17
                                                                                                                           18
18
                                                                                                                                   always@(/*AUTOSENSE*/sq\_sx\_exp\_number\ or\ sq\_sx\_exp\_type)
                                                                                                                           19
19
                                                                                                                           20
20
       //Write Pointer management for the pixel/poisition exports
                                                                                                                           21
                                                                                                                                        case(sq_sx_exp_type)
21
                                                                                                                           22
                                                                                                                                         2'b00: //pixels...no Z
22
       //alloc wr head ptr indicates the offset location of the next available space that
                                                                                                                           23
                                                                                                                                          begin
23
       //can be used by the next allocation request coming from the Sequencer
                                                                                                                           24
                                                                                                                                            case(sq_sx_exp_number)
24
       //Once a given valid export allocation request comes, the value of alloc_wr_head_ptr is
                                                                                                                           25
                                                                                                                                               2'b00: buffer size = 8'h4:
25
       //incemented by buffer size
                                                                                                                                               2'b01: buffer_size = 8'h8;
                                                                                                                                                                       Page 48 of 100
                                            Page 47 of 100
                                                                      Ex. 2106 - export control.v
                                                                                                                                                                                                 Ex. 2106 - export control.v
```

```
2'b10: buffer size = 8'hc;
                                                                                                                                                   end
 2
                                                                                                                              2
                    2'b11: buffer_size = 8'h10;
                                                                                                                                                  default
                 endcase // case(sq_sx_exp_number)
                                                                                                                                                    pos_buffer_size = 8'h4;
              2'b01: //pixels...with Z
                                                                                                                                                    no_aux_buffer = 1'b0;
                case(sq_sx_exp_number)
                                                                                                                                                  endcase // case(sq_sx_exp_number)
                   2'b00:buffer_size = 8'h8;
                   2'b01:buffer_size = 8'hc;
                   2'b10:buffer_size = 8'h10;
11
                    2'b11:buffer_size = 8'h14;
                                                                                                                                             case(sq_sx_exp_number)
12
                 endcase // case(sq_sx_exp_number)
13
                                                                                                                                                2'b01:;
14
15
                                                                                                                             15
                                                                                                                                                2'b11:;
              begin
16
                                                                                                                             16
                 case(sq_sx_exp_number)
                                                                                                                                                 endcase // case(sq_sx_exp_number)
17
                   2'b00:
                                                                                                                             17
                                                                                                                             18
18
                     begin
                                                                                                                                          endcase // case(sq_sx_exp_type)
19
                      pos_buffer_size = 8'h4;
                                                                                                                             19
                                                                                                                                      end // always@ (...
20
                      no_aux_buffer = 1'b1;
                                                                                                                             20
21
                     end
                                                                                                                             21
                                                                                                                                     wire [2:0] colors_present;
22
                    2'b01:
                                                                                                                             22
                                                                                                                                     //assign colors_present = current_attr_state[3:1];
23
                                                                                                                             23
                      pos_buffer_size = 8'h8;
24
                                                                                                                             24
                                                                                                                                     assign colors_present = 2'b10;
                       no_aux_buffer = 1'b0;
                                                                                                                                     wire [0:0] z_present;
                                             Page 49 of 100
                                                                                                                                                                           Page 50 of 100
                                                                       Ex. 2106 - export_control.v
                                                                                                                                                                                                    Ex. 2106 - export_control.v
                                                                                                                                     assign ring_wrapped = alloc_wr_head_ptr > alloc_wr_tail_ptr;
       assign z_present = 1'b0;
                                                                                                                                   assign \qquad space\_avail = (ring\_wrapped) \ ? \ BUFFER\_SIZE \ - \ alloc\_wr\_head\_ptr \ + \\ alloc\_wr\_tail\_ptr : alloc\_wr\_tail\_ptr - alloc\_wr_head\_ptr;
 2
       wire [2:0] pixel_size = colors_present + z_present;
               ql_exp_valid;
                                                                                                                                    //--Position Export write pointer managment
 6
       always @(posedge sclk)
 7
 8
            if(srst)
                                                                                                                             10
                                                                                                                                    //increment vs. decrement logic for the available position data in
10
               alloc_wr_head_ptr <= 7'b0;
                                                                                                                                    //the position count
11
              alloc_wr_tail_ptr <= BUFFER_SIZE;
                                                                                                                                    wire [1:0] pos_inc_dec;
12
               q1\_exp\_valid <= 1"b0;
13
14
                                                                                                                                    //01: decrement the count of remaining position data in the position buffer
15
                                                                                                                                    //10: increment count of the available position data in the position buffer
16
             if(sq_sx_exp_valid & exporting_pixel)
                                                                                                                             17
                                                                                                                                    //11: do nothing
17
                begin
                                                                                                                             18
                                                                                                                                    assign pos_inc_dec = {exported_pos_inc,exported_pos_dec};
18
                     alloc_wr_head_ptr <= alloc_wr_head_ptr + buffer_size;
                                                                                                                             19
19
                                                                                                                             20
                                                                                                                                    wire pos_outstanding;
                q1\_exp\_valid <= q\_exp\_valid;
20
                                                                                                                             21
                                                                                                                                     assign \quad pos\_outstanding = (exported\_pos\_count) \ ? \ 1'b1 : 1'b0;
21
              end // else: !if(srst)
                                                                                                                             22
22
        end // always @ (posedge sclk)
                                                                                                                             23
24
                                                                                                                                  assign \quad clipp\_outstanding\_req = pos\_outstanding \& pa\_pos\_req; \ //there's \ a position \ request from Clipper/PA
23
24
       wire [0:0] ring_wrapped;
                                                                                                                             25
       wire [6:0] space_avail;
                                                                                                                                     //two separate counters,one for each position thread
                                             Page 51 of 100
                                                                                                                                                                          Page 52 of 100
                                                                       Ex. 2106 - export control.v
                                                                                                                                                                                                    Ex. 2106 - export control.v
```

```
//increment the counter if we have a position export coming down from the shader pipe
2
3
     always @(posedge sclk)
                                                                                                                 //use to increment the position count
4
      begin
                                                                                                                 exported\_pos\_inc <= (\sim q\_exp\_pix\_alu1 \ \& \ q0\_valid\_export) \ ? \ 1:0;
          if(srst)
                                                                                                                end // else: !if(~no_aux_buffer)
           pos0_exp_count <= 2'b0;
           pos1_exp_count <= 2'b0;
                                                                                                            begin
           q_position_aux <= 1'b0;
                                                                                                               if(~no_aux_buffer)
           q_attribute_offset <= 2'b0;
                                                                                               11
                                                                                                                begin
11
                                                                                                   12
                                                                                               13
                                                                                                   13
           q_position_aux <= position_aux;
                                                                                               16
                                                                                                                end
15
            q attribute offset <= attribute offset;
                                                                                               17
                                                                                                               else
16
            if(q_sp0_exp_alu_id)
                                                                                               18
17
             begin
                                                                                                   18
                if(~no_aux_buffer)
19
                begin
    20
21
                                                                                               22
                                                                                                                 //use to increment the position count
    23
                                                                                                                 exported_pos_inc <= (~q_exp_pix_alu0 & q0_valid_export) ? 1: 0;
                                                                                               24
                                                                                                                end // else: !if(~no aux buffer)
24
                end
                                                                                               25
                                                                                                            end
25
                else
                                                                                               26
                                                                                                          end // else: !if(srst)
                 begin
                                                                                                      end // always @ (posedge sclk)
                                  Page 53 of 100
                                                                                                                                  Page 54 of 100
                                                      Ex. 2106 - export_control.v
                                                                                                                                                     Ex. 2106 - export_control.v
                                                                                                         if((q\_pos0\_exp\_count == 2'b11)\&(\sim pos\_buff0\_ready)) \\
                                                                                                2
                                                                                                          pos buff0 ready <= 1'b1;
                                                                                                          pos buff0 ready <= 1'b0;
     //not sure on whether we need these registered versions of count ANDI ???
                                                                                                         if((q\_pos1\_exp\_count == 2"b11)\&(\sim pos\_buff1\_ready))
     always @(posedge sclk)
7
                                                                                                          pos buff1 ready <= 1'b1;
          if(srst)
                                                                                                          pos_buff1_ready <= 1'b0;
10
           q_pos0_exp_count <= 2'b0;
                                                                                               10
                                                                                                      end // always @ (posedge sclk)
11
           q\_pos1\_exp\_count <= 2"b0;
                                                                                               11
12
                                                                                               12
13
                                                                                               13
                                                                                                     assign pos_data_ready = pos_buff0_ready & pos_buff1_ready;
14
                                                                                               14
15
                                                                                               15
           q_pos0_exp_count <= pos0_exp_count;
16
            q_pos1_exp_count <= pos1_exp_count;
                                                                                               16
                                                                                                     always @(posedge sclk)
17
                                                                                               17
           end
                                                                                                      begin
18
                                                                                               18
                                                                                                         if(srst)
19
                                                                                                          begin
20
                                                                                               20
     //this flags may potentially be used in the logic
                                                                                                           exported pos count <= 7'b0;
21
     //but for now they only have a debug purpose
                                                                                               21
                                                                                                          end
22
                                                                                               22
     reg \quad pos\_buff0\_ready \ , pos\_buffl\_ready; \\
                                                                                                         else
23
                                                                                               23
                                                                                                          begin
24
                                                                                               24
     always @(posedge sclk)
                                                                                                           case(pos_inc_dec)
25
       begin
                                                                                               25
                                                                                                           2'b00:; //do nothing
                                  Page 55 of 100
                                                                                                                                 Page 56 of 100
                                                      Ex. 2106 - export control.v
                                                                                                                                                     Ex. 2106 - export control.v
```

```
2'b01: exported_pos_count <= exported_pos_count - 1;
 2
                 2"b10: exported\_pos\_count <= exported\_pos\_count + 1;\\
                                                                                                                                    reg [0:0] oexp_count_rdy;
                2'b11:: //do nothing
                                                                                                                                    reg [0:0] oexp_pos_avail;
                endcase // case(pos_inc_dec)
                                                                                                                                    reg [6:0] oexp_buff_avail;
              end // else: !if(srst)
                                                                                                                                    //registering the outputs
        always @(posedge sclk)
                                                                                                                                    always @(posedge sclk)
        begin
10
                                                                                                                             10
                                                                                                                                          oexp_count_rdy <= q_exp_valid;
11
                                                                                                                            11
                                                                                                                                         oexp_buff_avail <= space_avail;
               pos_wr_head_ptr <= 4'h0; // location 80 ...the upper 16 locations reserved for
                                                                                                                            12
      position data
13
                                                                                                                            13
14
                pos_wr_tail_ptr <= 4'hf; //last location in the position export buffer
                                                                                                                                    assign sx sq exp count rdy = oexp count rdy;
15
              end
                                                                                                                            15
                                                                                                                                    assign sx_sq_exp_buf_avail = oexp_buff_avail;
16
            else
                                                                                                                            16
17
                                                                                                                            17
18
               if(sq_sx_exp_valid & exporting_position) //exporting vertex positions
                                                                                                                            18
19
                                                                                                                            19
                                                                                                                                    //State/rbbm bus decoding logic---
20
                     pos\_wr\_head\_ptr <= pos\_wr\_head\_ptr + pos\_buffer\_size;
                                                                                                                            20
21
                                                                                                                            21
22
              end // else: !if(srst)
                                                                                                                            22
23
        end // always @ (posedge sclk)
                                                                                                                            23
24
                                                                                                                            24
                                                                                                                                   // SQ_IMPORTS_EXPORTS
                                                                                                                                                                     <GFXDEC0:0x043C> 32 {
25
                                                                                                                            25
                                                                                                                                                                          PS_EXPORT_MODE 4:0 NUM DEF=2;
       //----Export buffer status output interface to SQ---
                                            Page 57 of 100
                                                                                                                                                                          Page 58 of 100
                                                                       Ex. 2106 - export_control.v
                                                                                                                                                                                                   Ex. 2106 - export_control.v
                                             VS_EXPORT_MODE 9:8 NUM DEF=0;
                                                                                                                                  //{ ALPHA_REF      "Reference value for alpha test, which is specified in IEEE floating point but stored in the RB internal format and converted back to IEEE floating point on reads. Therefore, this register may read back as a different value.";
 2
                                           PARAM_GEN_I0 12:12 ALPHA {
                                                                  "Take parameter 0 from the
                                                                                                                             4
      Parameter Cache.", //
Parameter 0."
                                            } DEF=0;
                                                                                                                              6 // RB_ALPHA_REF <GFXDEC0:0x081C> 32 FA {
       //
                                 GEN_INDEX 16:16 ALPHA {
                                                                                                                              7 // ALPHA_REF 31:0 DATA_TYPE="float";
                                                                   "Do not auto generate index
                                                                 "Auto generate index adresses."
12
13
      // SQ_IMPORTS_EXPORTS "Import export control"
14
      // {
       // PS_EXPORT_MODE "Pixel Shader exporting mode\n. 0xxxx: Normal mode\n. 1xxxx:
                                                                                                                            15
16
      Multipass.\n If normal,
                                                                                                                                    //GFX decode space values
                                                                                                                            16
                      bbbz where bbb is how many color we export (0-4) and z is export z or not.\n
18
      If multipass mode, 1-12 exports for color.
                                                                                                                            17
                                                                                                                                    parameter [3:0] gfxdec0 = 4h8,
      // VS_EXPORT_MODE "0: Position (1 vector).\n 1: Position (2 vectors).\n 2: Multipass
                                                                                                                            18
                                                                                                                                                  gfxdec1 = 4'h9
                                                                                                                                                   gfxdec2 = 4'ha,
                                                                                                                            19
21
22
       // PARAM_GEN_I0
                                   "Do we overwrite or not parameter 0 with generated XYST or
                                                                                                                            20
                                                                                                                                                   gfxdec3 = 4'hb,
      // GEN_INDEX "Auto generates an address from 0 to XX. Puts the results into R0 or 1 for pixels shaders and R2 for vertex shaders";
23
24
                                                                                                                            21
                                                                                                                                                   gfxdec4 = 4'hc,
25
                                                                                                                            22
                                                                                                                                                   gfxdec5 = 4'hd
      // };
                                                                                                                            23
                                                                                                                                                   gfxdec6 = 4'he,
26
                                                                                                                            24
                                                                                                                                                   gfxdec7 = 4'hf;
27
    // RB_ALPHA_REF ""
                                                                                                                                    // register offsets
                                                                                                                                                                          Page 60 of 100
                                             Page 59 of 100
                                                                       Ex. 2106 - export control.v
                                                                                                                                                                                                   Ex. 2106 - export control.v
```

```
wire [14:0] alpha test2 a;
       parameter [9:0] sq_imports_exports = 10'h10F;
 2
        parameter [9:0] rb_alpha_ref = 10'h207;
                                                                                                                                   wire [14:0]
                                                                                                                                                alpha test3 a;
        parameter [9:0] rb alpha test = 10'h207;
                                                                                                                                   wire [14:0]
                                                                                                                                                alpha test4 a;
                                                                                                                                   wire [14:0]
                                                                                                                                                 alpha_test5_a;
                                                                                                                                   wire [14:0]
                                                                                                                                                 alpha_test6_a;
        wire [14:0] import_export0_a;
                                                                                                                                   wire [14:0]
                                                                                                                                                 alpha_test7_a;
        wire [14:0] import_export1_a;
                      import_export2_a;
        wire [14:0]
                    import_export3_a;
                                                                                                                                   //Absolute memory mapped registers for the state registers
        wire [14:0]
                                                                                                                           10
                    import_export4_a;
                                                                                                                                                 import\_export0\_a = \{gfxdec0, sq\_imports\_exports\};
                                                                                                                                   assign
11
        wire [14:0]
                    import_export5_a;
                                                                                                                           11
                                                                                                                                                 import_export1_a = {gfxdec1,sq_imports_exports};
                                                                                                                                   assign
12
        wire [14:0]
                    import_export6_a;
                                                                                                                           12
                                                                                                                                   assign
                                                                                                                                                 import_export2_a = {gfxdec2,sq_imports_exports};
13
        wire [14:0]
                                                                                                                           13
                      import export7 a;
                                                                                                                                   assign
                                                                                                                                                 import export3 a = {gfxdec3,sq imports exports};
14
                                                                                                                           14
                                                                                                                                                 import export4 a = {gfxdec4,sq imports exports};
                                                                                                                                   assign
15
       wire [14:0] alpha ref0 a;
                                                                                                                           15
                                                                                                                                                 import export5 a = {gfxdec5,sq imports exports};
                                                                                                                                   assign
16
       wire [14:0]
                      alpha refl a:
                                                                                                                           16
                                                                                                                                   assign
                                                                                                                                                 import export6 a = {gfxdec6,sq imports exports};
17
       wire [14:0]
                                                                                                                           17
                      alpha ref2 a;
                                                                                                                                   assign
                                                                                                                                                 import_export7_a = {gfxdec7,sq_imports_exports};
18
       wire [14:0]
                      alpha ref3 a;
                                                                                                                           18
19
       wire [14:0]
                      alpha ref4 a;
                                                                                                                           19
                                                                                                                                   assign
                                                                                                                                               alpha ref0 a = {gfxdec0,rb alpha ref};
20
       wire [14:0]
                      alpha ref5 a;
                                                                                                                           20
                                                                                                                                   assign
                                                                                                                                               alpha refl a = {gfxdecl,rb alpha ref};
21
       wire [14:0]
                      alpha ref6 a;
                                                                                                                           21
                                                                                                                                   assign
                                                                                                                                               alpha\_ref2\_a = \{gfxdec2,rb\_alpha\_ref\};
22
       wire [14:0]
                      alpha_ref7_a;
                                                                                                                           22
                                                                                                                                   assign
                                                                                                                                               alpha\_ref3\_a = \{gfxdec3,rb\_alpha\_ref\};
23
                                                                                                                           23
                                                                                                                                   assign
                                                                                                                                               alpha ref4 a = {gfxdec4,rb alpha ref};
24
       wire [14:0] alpha_test0_a;
                                                                                                                           24
                                                                                                                                   assign
                                                                                                                                               alpha_ref5_a = {gfxdec5,rb_alpha_ref};
25
       wire [14:0]
                      alpha_test1_a;
                                                                                                                           25
                                                                                                                                   assign
                                                                                                                                               alpha_ref6_a = {gfxdec6,rb_alpha_ref};
                                            Page 61 of 100
                                                                                                                                                                        Page 62 of 100
                                                                      Ex. 2106 - export_control.v
                                                                                                                                                                                                 Ex. 2106 - export_control.v
                    alpha_ref7_a = {gfxdec7,rb_alpha_ref};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state alpha ref5;
        assign
 2
                                                                                                                                   reg [31:0]
                                                                                                                                                 state alpha ref6;
       assign
                      alpha test0 a = {gfxdec0,rb alpha test};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_ref7;
 4
                      alpha test1 a = {gfxdec1.rb alpha test};
       assign
        assign
                      alpha test2 a = {gfxdec2,rb alpha test};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test0;
        assign
                      alpha_test3_a = {gfxdec3,rb_alpha_test};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test1;
        assign
                      alpha\_test4\_a = \{gfxdec4,rb\_alpha\_test\};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test2;
        assign
                      alpha\_test5\_a = \{gfxdec5,rb\_alpha\_test\};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test3;
        assign
                      alpha\_test6\_a = \{gfxdec6,rb\_alpha\_test\};
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test4;
10
        assign
                      alpha\_test7\_a = \{gfxdec7,rb\_alpha\_test\};
                                                                                                                           10
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test5;
11
                                                                                                                           11
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test6;
12
                                                                                                                           12
                                                                                                                                   reg [31:0]
                                                                                                                                                 state_alpha_test7;
13
                                                                                                                           13
14
                                                                                                                           14
        //State Registers
15
                                                                                                                           15
                                                                                                                                   always @(posedge sclk)
16
     // reg [31:0] state_import_export0,state_import_export1;
                                                                                                                           16
                                                                                                                                    begin
17
                                                                                                                           17
                                                                                                                                        if(state_soft_reset)
     // reg [31:0]
                    state_import_export2,state_import_export3;
18
     // reg [31:0] state import export4,state import export5;
                                                                                                                           18
                                                                                                                                         begin
19
     // reg [31:0] state_import_export6,state_import_export7;
                                                                                                                                           state_import_export0 <= 32'b0;
20
                                                                                                                           20
                                                                                                                                           state import export1 <= 32'b0;
21
       reg [31:0]
                     state alpha ref0:
                                                                                                                           21
                                                                                                                                           state import export2 <= 32'b0;
22
                                                                                                                           22
       reg [31:0]
                      state alpha refl:
                                                                                                                                           state import export3 <= 32'b0:
23
                                                                                                                           23
       reg [31:0]
                      state alpha ref2:
                                                                                                                                           state import export4 <= 32'b0:
24
                                                                                                                           24
       reg [31:0]
                      state alpha ref3:
                                                                                                                                           state import export5 <= 32'b0;
25
       reg [31:0]
                      state_alpha_ref4;
                                                                                                                                           state_import_export6 <= 32'b0;
                                            Page 63 of 100
                                                                                                                                                                        Page 64 of 100
                                                                      Ex. 2106 - export control.v
                                                                                                                                                                                                 Ex. 2106 - export control.v
```

```
state import export7 <= 32'b0;
                                                                                                                                                import_export3_a:state_import_export3 <= state_wd;
 2
                state alpha ref0 <= 32'b0;
                                                                                                                                                import\_export4\_a:state\_import\_export4 <= state\_wd;
                state alpha ref1 <= 32'b0;
                                                                                                                                                import\_export5\_a:state\_import\_export5 <= state\_wd;
                state_alpha_ref2 <= 32'b0;
                                                                                                                                                import_export6_a:state_import_export6 <= state_wd;
                state_alpha_ref3 <= 32'b0;
                                                                                                                                                import\_export7\_a:state\_import\_export7 <= state\_wd;
                state_alpha_ref4 <= 32'b0;
                                                                                                                                                alpha\_ref0\_a: state\_alpha\_ref0 <= state\_wd;
                state_alpha_ref5 <= 32'b0;
                                                                                                                                                alpha_refl_a : state_alpha_refl <= state_wd;
                state_alpha_ref6 <= 32'b0;
                                                                                                                                                alpha_ref2_a : state_alpha_ref2 <= state_wd;
                state_alpha_ref7 <= 32'b0;
                                                                                                                                                alpha_ref3_a: state_alpha_ref3 <= state_wd;
10
                state_alpha_test0 <= 32'b0;
                                                                                                                               10
                                                                                                                                                alpha\_ref4\_a: state\_alpha\_ref4 <= state\_wd
11
                state_alpha_test1 <= 32'b0;
                                                                                                                                                alpha_ref5_a : state_alpha_ref5 <= state_wd;
12
                state_alpha_test2 <= 32'b0;
                                                                                                                                                alpha\_ref6\_a: state\_alpha\_ref6 <= state\_wd;
13
                state_alpha_test3 <= 32'b0;
                                                                                                                                                alpha_ref7_a : state_alpha_ref7 <= state_wd;
                state_alpha_test4 <= 32'b0;
                                                                                                                                                alpha_test0_a: state_alpha_test0 <= state_wd;
15
                state alpha test5 <= 32'b0;
                                                                                                                               15
                                                                                                                                                alpha test1 a : state alpha test1 <= state wd;
                state alpha test6 <= 32'b0:
16
                                                                                                                               16
                                                                                                                                                alpha test2 a : state alpha test2 <= state wd:
17
                state alpha test7 <= 32'b0:
                                                                                                                               17
                                                                                                                                                alpha test3 a : state alpha test3 <= state wd:
18
                                                                                                                               18
                                                                                                                                                alpha test4 a : state alpha test4 <= state wd;
19
                                                                                                                               19
                                                                                                                                                alpha test5 a : state alpha test5 <= state wd;
20
             else if(state we)
                                                                                                                               20
                                                                                                                                                alpha test6 a : state alpha test6 <= state wd;
21
              begin
                                                                                                                              21
                                                                                                                                                alpha test7 a: state alpha test7 <= state wd;
22
                case(state a)
                                                                                                                              22
                                                                                                                                               endcase // case(state a)
23
                 import_export0_a:state_import_export0 <= state_wd;
                                                                                                                              23
                                                                                                                                             end // if (state we)
24
                 import_export1_a:state_import_export1 <= state_wd;
                                                                                                                              24
                                                                                                                                        end // always @ (posedge sclk)
25
                 import_export2_a:state_import_export2 <= state_wd;
                                                                                                                              25
                                              Page 65 of 100
                                                                                                                                                                             Page 66 of 100
                                                                        Ex. 2106 - export_control.v
                                                                                                                                                                                                       Ex. 2106 - export_control.v
 2
                                                                                                                                2
                                                                                                                                              begin
        //Select Logic used to decide which contex will be used at any given time (selecting between
                                                                                                                                                state alpha test = state alpha test0;
      state(s) 0-7)
                                                                                                                                                state alpha ref = state alpha ref0;
                                                                                                                                                state_import_export = state_import_export0;
                                                                                                                                6
                                                                                                                                              end
        reg [31:0] state_alpha_test,state_alpha_ref,state_import_export;
                                                                                                                                             3'h1-
                                                                                                                                7
                                                                                                                                              begin
                                                                                                                                                state_alpha_test = state_alpha_test1;
10
        assign current_exp_state = (sp0_sx_exp_alu_id)? q_exp_state_alu1: q_exp_state_alu0;
                                                                                                                               10
                                                                                                                                                state_alpha_ref = state_alpha_ref1;
11
                                                                                                                              11
                                                                                                                                                state_import_export = state_import_export1;
12
        always @(/*AUTOSENSE*/current_exp_state or state_alpha_ref0
                                                                                                                               12
                                                                                                                                              end
13
               or state_alpha_ref1 or state_alpha_ref2
                                                                                                                               13
                                                                                                                                             3'h2:
14
               or state_alpha_ref3 or state_alpha_ref4
                                                                                                                               14
15
               or state_alpha_ref5 or state_alpha_ref6
                                                                                                                               15
                                                                                                                                                state_alpha_test = state_alpha_test2;
16
               or state_alpha_ref7 or state_alpha_test0
                                                                                                                                                state_alpha_ref = state_alpha_ref2;
17
               or state_alpha_test1 or state_alpha_test2
                                                                                                                                                state_import_export = state_import_export2;
18
               or state_alpha_test3 or state_alpha_test4
19
               or state alpha test5 or state alpha test6
               or state alpha test7 or state import export0
20
                                                                                                                              20
21
               or state import export1 or state import export2
                                                                                                                              21
                                                                                                                                               state alpha test = state alpha test3;
22
               or state_import_export3 or state_import_export4
                                                                                                                              22
                                                                                                                                                state alpha ref = state alpha ref3:
23
               or state_import_export5 or state_import_export6
                                                                                                                              23
                                                                                                                                               state_import_export = state_import_export3;
24
               or state_import_export7)
                                                                                                                              24
                                                                                                                                              end
25
         begin
                                                                                                                              25
                                                                                                                                             3'h4
26
             case(current_exp_state)
                                              Page 67 of 100
                                                                                                                                                                            Page 68 of 100
                                                                        Ex. 2106 - export control.v
                                                                                                                                                                                                      Ex. 2106 - export control.v
```

```
begin
                                                                                                                                                                                                                                                      state_alpha_test = state_alpha_test0;
  2
                             state\_alpha\_test = state\_alpha\_test4;
                                                                                                                                                                                                                                                      state\_alpha\_ref = state\_alpha\_ref0;
                             state_alpha_ref = state_alpha_ref4;
                                                                                                                                                                                                                                                      state_import_export = state_import_export0;
                             state_import_export = state_import_export4;
                                                                                                                                                                                                                                              endcase // case(q_exp_state)
                        3'h5
                                                                                                                                                                                                                                       end // always @ (...
                             state_alpha_test = state_alpha_test5;
                            state_alpha_ref = state_alpha_ref5;
 10
                                                                                                                                                                                                                                     //decoding the "attributes per pixel" state for each alu thread.
                             state_import_export = state_import_export5;
11
                                                                                                                                                                                                                       11
12
                                                                                                                                                                                                                        12
                                                                                                                                                                                                                                    reg [3:0] attr_state; //attribute state
13
                                                                                                                                                                                                                       13
                                                                                                                                                                                                                                     always @(/*AUTOSENSE*/sq_sx_exp_state or state_import_export0
                           begin
14
                                                                                                                                                                                                                        14
                                                                                                                                                                                                                                                  or state_import_export1 or state_import_export2
                            state alpha test = state alpha test6;
15
                            state alpha ref = state alpha ref6;
                                                                                                                                                                                                                       15
                                                                                                                                                                                                                                                  or state import export3 or state import export4
16
                                                                                                                                                                                                                        16
                            state import export = state import export6;
                                                                                                                                                                                                                                                  or state import export5 or state import export6
17
                                                                                                                                                                                                                       17
                          end
                                                                                                                                                                                                                                                  or state import export7)
                        3'h7-
18
                                                                                                                                                                                                                       18
                                                                                                                                                                                                                                      begin
19
                          begin
                                                                                                                                                                                                                        19
                                                                                                                                                                                                                                              case(sq sx exp state)
20
                             state\_alpha\_test = state\_alpha\_test7;
                                                                                                                                                                                                                       20
                                                                                                                                                                                                                                                3'h0: attr\_state = state\_import\_export0[3:0]; //bbbz (0-4 \ color \ attributes + z \ attribute)
21
                            state alpha ref = state alpha ref7;
                                                                                                                                                                                                                       21
                                                                                                                                                                                                                                                3'h1:attr\_state = state\_import\_export1[3:0];
22
                            state_import_export = state_import_export7;
                                                                                                                                                                                                                       22
                                                                                                                                                                                                                                                3'h2:attr_state = state_import_export2[3:0];
23
                                                                                                                                                                                                                       23
                                                                                                                                                                                                                                                3'h3:attr_state = state_import_export3[3:0];
24
                        default
                                                                                                                                                                                                                       24
                                                                                                                                                                                                                                                3'h4:attr_state = state_import_export4[3:0];
                                                                                                                                                                                                                       25
                                                                                                                                                                                                                                                 3'h5:attr_state = state_import_export5[3:0];
                                                                              Page 69 of 100
                                                                                                                                                                                                                                                                                                      Page 70 of 100
                                                                                                                           Ex. 2106 - export_control.v
                                                                                                                                                                                                                                                                                                                                                   Ex. 2106 - export_control.v
                                                                                                                                                                                                                                            alpha_color_test u01_alpha_test( .mask_out(mask_pix01), .pixel_discard(discard01),
                        3'h6:attr state = state import export6[3:0];
                                                                                                                                                                                                                                   .data_in(sp0_sx_data1), .alpha_test(state_alpha_test[2:0]),
                        3'h7:attr state = state import export7[3:0];
                                                                                                                                                                                                                                   // .alpha_test_enable(state_alpha_test[3]),
.alpha_to_mask_enable(state_alpha_test[4]), .alpha_ref(state_alpha_ref),
                        default:attr_state = state_import_export0[3:0];
                                                                                                                                                                                                                                  . discard_color_func(state_alpha_test[9:8]), discard_alpha_func(state_alpha_test[11:10]), .sclk(sclk));
                      endcase // case(sq_sx_exp_state)
               end // always @ (...
              assign attr_count = attr_state[3:1] + attr_state[0:0];
                                                                                                                                                                                                                                     //second pixel of quad0
                                                                                                                                                                                                                                     // alpha_color_test u02_alpha_test( .mask_out(mask_pix02), .pixel_discard(discard02),
                                                                                                                                                                                                                                   .data_in(sp0_sx_data2), .alpha_test(state_alpha_test[2:0]),
                                                                                                                                                                                                                        11
                                                                                                                                                                                                                                   10
                                                                                                                                                                                                                        13
11
                                                                                                                                                                                                                                   . discard\_color\_func(state\_alpha\_test[9:8]), discard\_alpha\_func(state\_alpha\_test[11:10]), .sclk(sclk));
12
              wire [0:0] mask_pix00, mask_pix01,mask_pix02,mask_pix03;
13
              wire [0:0] discard00, discard01, discard02, discard03;
                                                                                                                                                                                                                        17
14
                                                                                                                                                                                                                        18
                                                                                                                                                                                                                                     //second pixel of quad0
15
                                                                                                                                                                                                                                   // alpha_color_test u03_alpha_test( .mask_out(mask_pix03), .pixel_discard(discard03), .data_in(sp0_sx_data3), .alpha_test(state_alpha_test[2:0]),
                                                                                                                                                                                                                        20
16
            //FIRST QUAD data processing logic
                                                                                                                                                                                                                                   \label{lem:condition} $$ .alpha\_test\_enable(state\_alpha\_test[3]), $$ .alpha\_to\_mask\_enable(state\_alpha\_test[4]), .alpha\_ref(state\_alpha\_ref), $$ .alpha\_test[3]), $$ .alpha\_test\_alpha\_test[4]), .alpha\_test\_alpha\_ref), $$ .alpha\_test\_alpha\_test[4]), .alpha\_test\_alpha\_test\_alpha\_test[4]), .alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_alpha\_test\_al
17
                                                                                                                                                                                                                        22
18
            //first pixel of quad0
                                                                                                                                                                                                                       23
24
25
                                                                                                                                                                                                                                   .discard_color_func(state_alpha_test[9:8]),.discard_alpha_func(state_alpha_test[11:10]), .sclk(sclk));
                      alpha\_color\_test \quad u00\_alpha\_test (\quad .mask\_out(mask\_pix00), \quad .pixel\_discard(discard00), \\
20
           . data\_in(sp0\_sx\_data0), . alpha\_test(state\_alpha\_test[2:0]), \\
                                                                                                                                                                                                                        26
           \label{lem:constraint} $$ $/f$ alpha_test_alpha_test_3], $$ alpha_to_mask_enable(state_alpha_test_4], alpha_ref(state_alpha_ref), $$
21
22
                                                                                                                                                                                                                       27
23
24
25
                                                                                                                                                                                                                                     wire [0:0] mask_pix10, mask_pix11,mask_pix12,mask_pix13;
           wire [0:0] discard10, discard11, discard12, discard13;
26
                                                                                                                                                                                                                        30
27
              //second pixel of quad0
                                                                                                                                                                                                                        31
                                                                              Page 71 of 100
                                                                                                                                                                                                                                                                                                      Page 72 of 100
                                                                                                                           Ex. 2106 - export control.v
                                                                                                                                                                                                                                                                                                                                                  Ex. 2106 - export control.v
```

```
//SECOND QUAD data processing logic
 2
                                                                                                                                                                                                                                                                                              // .alpha_test_enable(state_alpha_test[3]), .alpha_to_mask_enable(state_alpha_test[4]), .alpha_ref(state_alpha_ref),
  3
                //first pixel of quad1
              \label{lem:color_test} $$ $ u10\_alpha\_test( .mask\_out(mask\_pix10), .pixel\_discard(discard10), .data\_in(sp0\_sx\_data0), .alpha\_test(state\_alpha\_test(2:0]), $$ $ u10\_alpha\_test(state\_alpha\_test(2:0]), $$ $ u10\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state\_alpha\_test(state
                                                                                                                                                                                                                                                                                              .discard_color_func(state_alpha_test[9:8]),.discard_alpha_func(state_alpha_test[11:10]),
                                                                                                                                                                                                                                                                                              .sclk(sclk)):
              // .alpha_test_enable(state_alpha_test[3]),
.alpha_to_mask_enable(state_alpha_test[4]), .alpha_ref(state_alpha_ref),
                                                                                                                                                                                                                                                                                                //export buffers instatiated two banks of 4 128x128 register files each
              .discard_color_func(state_alpha_test[9:8]),.discard_alpha_func(state_alpha_test[11:10]),
10
              sclk(sclk) ):
                                                                                                                                                                                                                                                                               10
                                                                                                                                                                                                                                                                                            reg [8:0] exp_read_pointer; //final read pointer in Export Buffers //8 bits for address + 1 bit for keep/discard auad
11
12
                //second pixel of quad1
                                                                                                                                                                                                                                                                               13
                                                                                                                                                                                                                                                                                                reg [8:0] q_exp_read_pointer; //registered version of the above
              \label{lem:color_test} $$ $ \  \, \text{ul1\_alpha\_test}( \ .mask\_out(mask\_pix11), \ .pixel\_discard(discard11), .data\_in(sp0\_sx\_data1), .alpha\_test(state\_alpha\_test(2:0]), $$ $$
13
14
                                                                                                                                                                                                                                                                               14
              // .alpha_test_enable(state_alpha_test[3]),
.alpha_to_mask_enable(state_alpha_test[4]), .alpha_ref(state_alpha_ref),
                                                                                                                                                                                                                                                                                15
                                                                                                                                                                                                                                                                                                reg exp buff read en;
                                                                                                                                                                                                                                                                                                            q_exp_buff_read_en;
                                                                                                                                                                                                                                                                                17
              .discard color func(state alpha test[9:8]),.discard alpha func(state alpha test[11:10]),
19
              .sclk(sclk) );
                                                                                                                                                                                                                                                                                18
                                                                                                                                                                                                                                                                                                //flags representing a read request from the 5 different clients into SX (Clipper, RB0-3)
20
                                                                                                                                                                                                                                                                               19
                                                                                                                                                                                                                                                                                                                   service_rb0, service_rb1, service_rb2, service_rb3, service_clipper;
21
                 //second pixel of quad1
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       [1:0]
              \label{lem:color_test} $$ $ \mu_2 = \frac{1}{2} alpha_test( .mask_out(mask_pix12), .pixel_discard(discard12), .data_in(sp0_sx_data2), .alpha_test(state_alpha_test(2:0]), $$ $$
                                                                                                                                                                                                                                                                               21
                                                                                                                                                                                                                                                                                           rb0_grant_count,rb1_grant_count,rb2_grant_count,rb3_grant_count,clipper_grant_count;
22
23
                                                                                                                                                                                                                                                                                             \label{eq:count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_count_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_grant_qrb2_gran
                                                                                                                                                                                                                                                                                22
23
              // .alpha_test_enable(state_alpha_test[3]),
.alpha_to_mask_enable(state_alpha_test[4]), .alpha_ref(state_alpha_ref),
                                                                                                                                                                                                                                                                               24
                                                                                                                                                                                                                                                                                                             read_valid_rb0, read_valid_rb1, read_valid_rb2, read_valid_rb3;
26
27
              .discard_color_func(state_alpha_test[9:8]),.discard_alpha_func(state_alpha_test0[11:10]),
                                                                                                                                                                                                                                                                               25
                                                                                                                                                                                                                                                                                                                  q\_read\_valid\_rb0, q\_read\_valid\_rb1, q\_read\_valid\_rb2, q\_read\_valid\_rb3;
28
              .sclk(sclk) );
                                                                                                                                                                                                                                                                               26
29
                                                                                                                                                                                                                                                                               27
                                                                                                                                                                                                                                                                                                                   read_valid_clipp, q_read_valid_clipp;
30
                //second pixel of quad1
                                                                                                                                                                                                                                                                                28
                                                                                                  Page 73 of 100
                                                                                                                                                                                                                                                                                                                                                                                  Page 74 of 100
                                                                                                                                                           Ex. 2106 - export_control.v
                                                                                                                                                                                                                                                                                                                                                                                                                                          Ex. 2106 - export_control.v
                 //bypassing the alpha compare and color compare 1.0/0.0 logic ....revisit this logic ANDI !!!!!
                                                                                                                                                                                                                                                                                                                                                               .imem re(q exp buff read en),
                                                                                                                                                                                                                                                                                             . iread\_valid\_rb0(q\_read\_valid\_rb0 \ \& \ q\_exp\_read\_pointer[8]), iread\_valid\_rb1(q\_read\_valid\_rb1 \ \& \ q\_exp\_read\_pointer[8]),
                 export buffers uexport buffers(
                                                                                                                                                                                                                                                                                             . iread\_valid\_rb2(q\_read\_valid\_rb2 \ \& \ q\_exp\_read\_pointer[8]), iread\_valid\_rb3(q\_read\_valid\_rb3 \ \& \ q\_exp\_read\_pointer[8]),
                                                                               .orb0 data(sx rb0 color data), .orb1 data(sx rb1 color data).
                                                                               .orb2 data(sx rb2 color data), .orb3 data(sx rb3 color data),
                                                                                                                                                                                                                                                                                                                                                               .iread_valid_clipp(q_read_valid_clipp)
              .orb0_data_valid(sx_rb0_color_send),.orb1_data_valid(sx_rb1_color_send),
                orb2_data_valid(sx_rb2_color_send),.orb3_data_valid(sx_rb3_color_send),
                                                                                                                                                                                                                                                                                 10
 10
                                                                               .oclipp_data(sx_pa_data), .oclipp_data_valid(sx_pa_send),
11
                                                                                                                                                                                                                                                                                                //----PA request position request interface skid buffer--
                                                                               .iread_addr(q_exp_read_pointer[7:0]),
                                                                               .iwrite addr(export index)
                                                                                                                                                                                                                                                                                15
16
15
                                                                               .ipixel data0(q0 sp0 data0),.ipixel data1(q0 sp0 data1),
                                                                                                                                                                                                                                                                                17
                                                                               .ipixel data2(q0 sp0 data2),.ipixel data3(q0 sp0 data3),
                                                                                                                                                                                                                                                                               18
                                                                                                                                                                                                                                                                                                //this the request packet from PA
17
                                                                               .ipixel data4(q0 sp1 data0),.ipixel data5(q0 sp1 data1),
                                                                                                                                                                                                                                                                                19
                                                                                                                                                                                                                                                                                                //pa_sp_id : which bank of SPs data should come from
18
                                                                               .ipixel_data6(q0_sp1_data2),.ipixel_data7(q0_sp1_data3),
                                                                                                                                                                                                                                                                               20
                                                                                                                                                                                                                                                                                                //pa_offset : 0-3 in the group of 4 position vectors
 20
               .iphase_rb0(q_rb0_grant_count),.iphase_rb1(q_rb1_grant_count),
                                                                                                                                                                                                                                                                                                 //pa_last: the last position request of this specific vector ...increment the read pointer by 1 and
21
22
                                                                                                                                                                                                                                                                               22
23
                iphase_rb2(q_rb2_grant_count),.iphase_rb3(q_rb3_grant_count),
                                                                                                                                                                                                                                                                                             free the space
23
                                                                               .iphase_clipp(q_clipp_grant_count),
                                                                                                                                                                                                                                                                               24
24
                                                                                                                                                                                                                                                                               25
                                                                                                                                                                                                                                                                                                assign \quad pa\_req\_packet = \{pa\_sx\_sp\_id, pa\_sx\_offset, pa\_sx\_aux, pa\_sx\_last\};
25
                                                                                                                                                                                                                                                                               26
26
                                                                               .imem_wen(export_buffer_wen),
                                                                                                                                                                                                                                                                               27
                                                                                                                                                                                                                                                                                                skid_buff_top #(5,8) pa_pos_req_buff(
                                                                               .imem_wew(export_buffer_wew),
                                                                                                                                                                                                                                                                                28
                                                                                                                                                                                                                                                                                                                                                                           write rts(pa sx req).
                                                                                                  Page 75 of 100
                                                                                                                                                                                                                                                                                                                                                                                 Page 76 of 100
                                                                                                                                                           Ex. 2106 - export control.v
                                                                                                                                                                                                                                                                                                                                                                                                                                         Ex. 2106 - export control.v
```

```
.write_rtr(sx_pa_req_rtr),
                                                                                                                                                                                                                                                                                                                                                       2
                                                                                                                   .write_data(pa_req_packet),
                                                                                                                                                                                                                                                                                                                                                                         wire
                                                                                                                                                                                                                                                                                                                                                                                               sx\_to\_rb0\_index\_rtr, sx\_to\_rb1\_index\_rtr, sx\_to\_rb2\_index\_rtr, sx\_to\_rb3\_index\_rtr;
                                                                                                                   .read_rts(pa_pos_req),
                                                                                                                                                                                                                                                                                                                                                                                               osx\_rb0\_index\_rtr, osx\_rb1\_index\_rtr, osx\_rb2\_index\_rtr, osx\_rb3\_index\_rtr;
                                                                                                                   .read_rtr(service_clipper),
                                                                                                                                                                                                                                                                                                                                                                                               rb0\_read\_req, rb1\_read\_req, rb2\_read\_req, rb3\_read\_req;
                                                                                                                   .read_data(pa_req_control),
                                                                                                                  .clk(sclk),
                                                                                                                                                                                                                                                                                                                                                                                               rb0\_color\_ready, rb1\_color\_ready, rb2\_color\_ready, rb3\_color\_ready;
                                                                                                                                                                                                                                                                                                                                                                    //there's an outstanding request for color and RB is ready to accept color data for the next four cycles % \left( 1\right) =\left( 1\right) \left( 1\right) \left(
                                                                                                                                                                                                                                                                                                                                                     10
                                                                                                                                                                                                                                                                                                                                                                        assign rb0_color_ready = rb0_read_req & rb0_sx_color_rtr;
                      always @(posedge sclk)
                                                                                                                                                                                                                                                                                                                                                     11
                                                                                                                                                                                                                                                                                                                                                                         assign rbl_color_ready = rbl_read_req & rbl_sx_color_rtr;
11
                        begin
                                                                                                                                                                                                                                                                                                                                                     12
                                                                                                                                                                                                                                                                                                                                                                         assign rb2_color_ready = rb2_read_req & rb2_sx_color_rtr;
12
                                                                                                                                                                                                                                                                                                                                                    13
                                                                                                                                                                                                                                                                                                                                                                         assign rb3_color_ready = rb3_read_req & rb3_sx_color_rtr;
13
                                      q_pa_req_control <= 5'b0;
                                                                                                                                                                                                                                                                                                                                                    14
14
                                  else if(service_clipper)
                                                                                                                                                                                                                                                                                                                                                    15
                                                                                                                                                                                                                                                                                                                                                                         wire [8:0] rb0 read index, rb1 read index, rb2 read index, rb3 read index;
15
                                      q_pa_req_control <= pa_pos_req;
                                                                                                                                                                                                                                                                                                                                                    16
                                                                                                                                                                                                                                                                                                                                                                         reg~[8:0]~~q\_rb0\_read\_index,q\_rb1\_read\_index,q\_rb2\_read\_index,q\_rb3\_read\_index;
16
                                                                                                                                                                                                                                                                                                                                                    17
17
                                                                                                                                                                                                                                                                                                                                                    18
                                                                                                                                                                                                                                                                                                                                                                         wire [8:0] rb0_index,rb1_index,rb2_index,rb3_index;
18
                                                                                                                                                                                                                                                                                                                                                    19
19
20
                                                                                                                                                                                                                                                                                                                                                    20
21
                    //Export Buffer Read logic
                                                                                                                                                                                                                                                                                                                                                    21
                                                                                                                                                                                                                                                                                                                                                                          assign exported_pos_dec = service_clipper;
22
23
                                                                                                                                                                                                                                                                                                                                                    22
                                                                                                                                                                                                                                                                                                                                                   23
                                                                                                                                                                                                                                                                                                                                                                                              rb0_index = {rb0_sx_index_op, rb0_sx_index};
24
                                                                                                                                                                                                                                                                                                                                                    24
                                                                                                                                                                                                                                                                                                                                                                                               rb1_index = {rb1_sx_index_op, rb1_sx_index};
25
                     //There are four skid buffers ..one for each rb on the index interface
                                                                                                                                                                                                                                                                                                                                                                                               rb2_index = {rb2_sx_index_op, rb2_sx_index};
                                                                                                                                                                                                                                                                                                                                                                                               rb3_index = {rb3_sx_index_op, rb3_sx_index};
                                                                                                                           Page 77 of 100
                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Page 78 of 100
                                                                                                                                                                                                  Ex. 2106 - export_control.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     Ex. 2106 - export_control.v
                                                                                                                                                                                                                                                                                                                                                                                          q_rb2_read_index <= rb2_read_index;
                                                                                                                                                                                                                                                                                                                                                       2
                                                                                                                                                                                                                                                                                                                                                                             end
                    //latching in the export read index for rb color requests
                                                                                                                                                                                                                                                                                                                                                                         always @(posedge sclk)
                     always @(posedge sclk)
                                                                                                                                                                                                                                                                                                                                                                           begin
                       begin
                                                                                                                                                                                                                                                                                                                                                                                         if(srst)
                                                                                                                                                                                                                                                                                                                                                                                             q_rb3_read_index <= 8'b0;
                                      q rb0 read index <= 8'b0;
                                                                                                                                                                                                                                                                                                                                                                                        else if(service rb3)
                                  else if(service_rb0)
                                                                                                                                                                                                                                                                                                                                                                                          q_rb3_read_index <= rb3_read_index;
10
                                      q\_rb0\_read\_index <= rb0\_read\_index;
                                                                                                                                                                                                                                                                                                                                                     10
11
                                                                                                                                                                                                                                                                                                                                                    11
12
                                                                                                                                                                                                                                                                                                                                                    12
13
                      always @(posedge sclk)
                                                                                                                                                                                                                                                                                                                                                                          skid_buff_top #(9,8) rb0_index_buff(
14
                                                                                                                                                                                                                                                                                                                                                                                                                                                             .write_rts(rb0_sx_index_send),
15
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .write_rtr(sx_to_rb0_index_rtr),
16
                                     q_rb1_read_index <= 8'b0;
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .write_data(rb0_index),
17
                                  else if(service_rb1)
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .read_rts(rb0_read_req),
18
                                      q_rb1_read_index <= rb1_read_index;
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .read_rtr(service_rb0),
19
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .read_data(rb0_read_index),
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .clk(sclk),
20
                                                                                                                                                                                                                                                                                                                                                     20
21
                                                                                                                                                                                                                                                                                                                                                    21
                                                                                                                                                                                                                                                                                                                                                                                                                                                              .reset(srst)
                    always @(posedge sclk)
22
                                                                                                                                                                                                                                                                                                                                                    22
                       begin
23
                                                                                                                                                                                                                                                                                                                                                    23
24
                                     q\_rb2\_read\_index <= 8"b0";
                                                                                                                                                                                                                                                                                                                                                    24
                                                                                                                                                                                                                                                                                                                                                                         skid_buff_top #(9,8) rb1_index_buff(
25
                                   else if(service rb2)
                                                                                                                                                                                                                                                                                                                                                    25
                                                                                                                                                                                                                                                                                                                                                                                                                                                               .write_rts(rb1_sx_index_send),
                                                                                                                           Page 79 of 100
                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Page 80 of 100
                                                                                                                                                                                                  Ex. 2106 - export control.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     Ex. 2106 - export control.v
```

```
.write_rtr(sx_to_rb1_index_rtr),
                                                                                                                                                                clk(sclk)
                                      .write data(rb1 index),
                                                                                                                                                                .reset(srst)
                                      .read rts(rb1 read req)
                                      .read_rtr(service_rb1),
                                      .read_data(rb1_read_index),
                                      .clk(sclk),
                                                                                                                                  //round robin arbriter (with a timer) to service read requests from each
                                      .reset(srst)
                                                                                                                                  //RB client into Export Buffers
        skid_buff_top #(9,8) rb2_index_buff(
                                     .write_rts(rb2_sx_index_send)
                                                                                                                          10
                                                                                                                                  reg [2:0] read_state, next_read_state;
11
                                     .write_rtr(sx_to_rb2_index_rtr),
                                                                                                                          11
                                                                                                                                        rb0_timeup, rb1_timeup, rb2_timeup, rb3_timeup;
12
                                     .write_data(rb2_index),
                                                                                                                          12
                                                                                                                                       clipper_timeup;
13
                                     .read_rts(rb2_read_req),
                                                                                                                          13
                                                                                                                                'define READ_IDLE 3'b000
                                      .read_rtr(service_rb2),
                                                                                                                          14
15
                                     .read_data(rb2_read_index),
                                                                                                                                'define GRANT_CLIPPER 3'b001
                                                                                                                          15
                                      .clk(sclk).
                                                                                                                                'define GRANT RB0 3'b010
16
                                                                                                                          16
17
                                                                                                                          17
                                                                                                                                'define GRANT_RB1 3'b011
                                      .reset(srst)
                                                                                                                          18
                                                                                                                                'define GRANT RB2 3'b100
18
19
        skid_buff_top #(9,8) rb3_index_buff(
                                                                                                                          19
                                                                                                                                'define GRANT RB3 3'b101
20
                                      .write rts(rb3 sx index send).
                                                                                                                          20
21
                                      .write rtr(sx to rb3 index rtr),
                                                                                                                          21
                                                                                                                                  always @(posedge sclk)
22
                                      .write_data(rb3_index),
                                                                                                                          22
23
                                      .read rts(rb3 read req),
                                                                                                                          23
24
                                      .read_rtr(service_rb3),
                                                                                                                          24
                                                                                                                                        read_state <= `READ_IDLE;
25
                                      .read_data(rb3_read_index),
                                                                                                                          25
                                            Page 81 of 100
                                                                                                                                                                       Page 82 of 100
                                                                      Ex. 2106 - export_control.v
                                                                                                                                                                                                Ex. 2106 - export_control.v
              read state <= next read state;
                                                                                                                                        end
2
                                                                                                                           2
        end
                                                                                                                                       else
 3
                                                                                                                                        begin
4
       wire clipp_count_inc;
                                                                                                                                         if(clipp_count_inc)
       wire rb0 count inc;
                                                                                                                                           clipper\_grant\_count <= clipper\_grant\_count + 1;
       wire rb1 count inc;
                                                                                                                                          if(rb0 count inc)
       wire rb2 count inc;
                                                                                                                                           rb0\_grant\_count <= rb0\_grant\_count + 1;
        wire rb3 count inc;
                                                                                                                                          if(rb1 count inc)
                                                                                                                                           rb1_grant_count <= rb1_grant_count + 1;
       assign clipp_count_inc = (clipper_grant_count != 2'b00) |service_clipper;
10
                                                                                                                          10
                                                                                                                                          if(rb2_count_inc)
11
        assign rb0_count_inc = (rb0_grant_count != 2'b00) |service_rb0;
                                                                                                                          11
                                                                                                                                           rb2\_grant\_count <= rb2\_grant\_count + 1;
12
        assign rb1_count_inc = (rb1_grant_count != 2'b00) |service_rb1;
                                                                                                                          12
                                                                                                                                          if(rb3_count_inc)
13
        assign rb2_count_inc = (rb2_grant_count != 2'b00) |service_rb2;
                                                                                                                                           rb3_grant_count <= rb3_grant_count + 1;
14
        assign rb3_count_inc = (rb3_grant_count != 2'b00) |service_rb3;
15
                                                                                                                          15
                                                                                                                                   end // always @ (posedge sclk)
16
       //time-out request service counter
17
       always @(posedge sclk)
                                                                                                                          17
18
        begin
                                                                                                                          18
                                                                                                                                  always @(posedge sclk)
19
            if(srst)
                                                                                                                          19
                                                                                                                                   begin
20
                                                                                                                          20
                                                                                                                                       if(srst)
21
               rb0 grant count <= 2'b0;
                                                                                                                          21
                                                                                                                                        begin
22
               rb1_grant_count <= 2'b0;
                                                                                                                          22
                                                                                                                                          q_rb0_grant_count <= 2'b0;
23
               rb2_grant_count <= 2'b0;
                                                                                                                          23
                                                                                                                                          q_rb1_grant_count <= 2'b0;
24
               rb3 grant count <= 2'b0;
                                                                                                                          24
                                                                                                                                          q_rb2_grant_count <= 2'b0;
               clipper_grant_count <= 2'b0;
                                                                                                                                          q\_rb3\_grant\_count <= 2"b0;
                                            Page 83 of 100
                                                                                                                                                                       Page 84 of 100
                                                                      Ex. 2106 - export control.v
                                                                                                                                                                                                Ex. 2106 - export control.v
```

```
q_clipp_grant_count <= 2'b0;
 2
               q\_exp\_read\_pointer <= 9"b0;
               q\_exp\_buff\_read\_en <= 1"b0;
               q_read_valid_clipp <= 1'b0;
                                                                                                                             always @(/*AUTOSENSE*/`GRANT_CLIPPER or `GRANT_RB0 or `GRANT_RB1
               q_read_valid_rb0 <= 1'b0;
                                                                                                                                    or `GRANT_RB2 or `GRANT_RB3 or `READ_IDLE
               q_read_valid_rb1 <= 1'b0;
                                                                                                                                    or clipp_outstanding_req or clipper_timeup
              q_read_valid_rb2 <= 1'b0;
                                                                                                                                    or rb0_color_ready or rb0_timeup or rb1_color_ready
              q_read_valid_rb3 <= 1'b0;
                                                                                                                                    or rb1_timeup or rb2_color_ready or rb2_timeup
                                                                                                                                    or rb3_color_ready or rb3_timeup or read_state)
10
                                                                                                                      10
11
                                                                                                                     11
                                                                                                                                  case(read_state)
12
               q\_rb0\_grant\_count <= rb0\_grant\_count;
                                                                                                                      12
                                                                                                                                   'READ_IDLE:
13
              q_rb1_grant_count <= rb1_grant_count;
                                                                                                                      13
                                                                                                                                    begin
14
               q_rb2_grant_count <= rb2_grant_count;
                                                                                                                                     if(clipp_outstanding_req & clipper_timeup)
                                                                                                                                         next_read_state = `GRANT_CLIPPER;
15
               q rb3 grant count <= rb3 grant count;
                                                                                                                      15
16
                                                                                                                      16
                                                                                                                                      else if(rb0 color ready & rb0 timeup)
               q clipp grant count <= clipper grant count;
17
                                                                                                                      17
                                                                                                                                         next read state = 'GRANT RB0:
               q exp read pointer <= exp read pointer;
18
               q_exp_buff_read_en <= exp_buff_read_en;
                                                                                                                      18
                                                                                                                                      else if(rb1 color ready & rb1 timeup)
19
               q read valid clipp <= read valid clipp;
                                                                                                                      19
                                                                                                                                         next_read_state = 'GRANT_RB1'
20
               q read valid rb0 <= read valid rb0;
                                                                                                                      20
                                                                                                                                      else if(rb2 color ready & rb2 timeup)
21
               q read valid rb1 <= read valid rb1;
                                                                                                                     21
                                                                                                                                         next read state = 'GRANT RB2:
22
               q\_read\_valid\_rb2 <= read\_valid\_rb2;
                                                                                                                     22
                                                                                                                                      else if(rb3 color ready & rb3 timeup)
23
               q read valid rb3 <= read valid rb3;
                                                                                                                     23
                                                                                                                                         next read state = 'GRANT RB3;
             end // else: !if(srst)
24
                                                                                                                     24
25
        end // always @ (posedge sclk)
                                                                                                                     25
                                                                                                                                         next_read_state = `READ_IDLE;
                                          Page 85 of 100
                                                                                                                                                                Page 86 of 100
                                                                   Ex. 2106 - export_control.v
                                                                                                                                                                                         Ex. 2106 - export_control.v
                                                                                                                                         next_read_state = `GRANT_RB3;
             'GRANT_CLIPPER:
                                                                                                                      2
 2
                                                                                                                                     else
              begin
                                                                                                                                         next read state = 'READ IDLE;
                if(clipp_outstanding_req & clipper_timeup)
                                                                                                                                    end
                   next read state = 'GRANT CLIPPER;
                                                                                                                                   'GRANT RB1:
                else if(rb0_color_ready & rb0_timeup)
                                                                                                                                    begin
                   next read state = 'GRANT RB0;
                                                                                                                                     if(clipp_outstanding_req & clipper_timeup)
                else if(rbl_color_ready & rbl_timeup)
                                                                                                                                         next read state = 'GRANT CLIPPER;
                   next_read_state = `GRANT_RB1;
                                                                                                                                      else if(rb2_color_ready)
                                                                                                                                         next_read_state = `GRANT_RB2;
10
                else if(rb2_color_ready & rb2_timeup)
                                                                                                                      10
11
                   next_read_state = `GRANT_RB2;
                                                                                                                     11
                                                                                                                                      else if(rb3_color_ready)
12
                else if(rb3_color_ready & rb3_timeup)
                                                                                                                      12
                                                                                                                                         next_read_state = `GRANT_RB3;
                   next_read_state = `GRANT_RB3;
13
                                                                                                                      13
                                                                                                                                      else if(rb0_color_ready)
14
                                                                                                                                         next_read_state = `GRANT_RB0;
15
                                                                                                                      15
                   next_read_state = `READ_IDLE;
16
                                                                                                                                         next_read_state = `READ_IDLE;
17
              GRANT_RB0:
18
                                                                                                                      18
                                                                                                                                   'GRANT_RB2:
              begin
19
                if(clipp outstanding req & clipper timeup)
                                                                                                                                    begin
20
                   next read state = 'GRANT CLIPPER;
                                                                                                                     20
                                                                                                                                     if(clipp outstanding req & clipper timeup)
21
                                                                                                                     21
                                                                                                                                         next read state = `GRANT CLIPPER;
                else if(rb1 color ready & rb1 timeup)
22
                                                                                                                     22
                   next read state = 'GRANT RB1:
                                                                                                                                      else if(rb3 color ready & rb3 timeup)
23
                                                                                                                     23
                                                                                                                                         next read state = 'GRANT RB3:
                else if(rb2 color ready & rb2 timeup)
24
                   next_read_state = 'GRANT_RB2'
                                                                                                                     24
                                                                                                                                      else if(rb0 color ready & rb0 timeup)
                else if(rb3 color ready & rb3 timeup)
                                                                                                                                         next read state = 'GRANT RB0;
                                          Page 87 of 100
                                                                                                                                                                Page 88 of 100
                                                                   Ex. 2106 - export control.v
                                                                                                                                                                                        Ex. 2106 - export control.v
```

```
else if(rb1 color ready & rb1 timeup)
 2
                    next read state = 'GRANT RB1;
                                                                                                                                     //pa_req_packet = {ipa_sp_id,ipa_offset,ipa_aux,ipa_last};
                 else
                                                                                                                                     wire [7:0] pos_exp_read_ptr;
                     next_read_state = `READ_IDLE;
                                                                                                                                     reg [3:0] pos_exp_read_offset;
              'GRANT_RB3:
                                                                                                                                     //creating the read pointer for clipper request
                if(clipp_outstanding_req & clipper_timeup)
                                                                                                                                    //pos_exp_read_offset : 4 entry buffer offset (possible values are 0,4,8,C)
                    next_read_state = `GRANT_CLIPPER;
                                                                                                                                     //pa_req_control[3:2] == pa_sx_offset at the interface ...an offset within the 4-entry buffer
10
                 else if(rb0_color_ready & rb0_timeup)
11
                   next_read_state = `GRANT_RB0;
                                                                                                                                    //last request for the current buffer
12
                else if(rb1_color_ready & rb1_timeup)
                                                                                                                                    //increment the pointer to the next buffer
13
                    next_read_state = `GRANT_RB1;
                                                                                                                             13
                                                                                                                                    wire pa req buff last = q pa req control[0];
14
                 else if(rb2_color_ready & rb2_timeup)
15
                    next read state = 'GRANT RB2;
                                                                                                                             15
                                                                                                                                    //updating pos exp read offset
16
                                                                                                                             16
                                                                                                                                    //depending on the mode of export the offset increase can be by 4 or 8
17
                                                                                                                             17
                    next read state = 'READ IDLE:
                                                                                                                                    //when an auxiliary buffer is present increment by 8
18
               end
                                                                                                                             18
                                                                                                                                    //when no auxiliary buffer ...increment by 4
19
              default:
                                                                                                                             19
                                                                                                                                     always @(posedge sclk)
20
                                                                                                                             20
21
                next read state = `READ IDLE:
                                                                                                                             21
                                                                                                                                          if(srst)
22
                                                                                                                             22
23
             endcase // case(read state)
                                                                                                                             23
                                                                                                                                            pos_exp_read_offset <= 4'b0;
24
        end // always @ (...
                                                                                                                             24
25
                                                                                                                             25
                                                                                                                                          else
                                             Page 89 of 100
                                                                                                                                                                           Page 90 of 100
                                                                       Ex. 2106 - export_control.v
                                                                                                                                                                                                    Ex. 2106 - export_control.v
 2
                                                                                                                              2
               if(pa req buff last) //ipa last...move to the next buffer
                                                                                                                                     assign pos_exp_read_ptr[7] = pa_req_sp_id; //which SP is the data coming from
                pos_exp_read_offset <= pos_exp_read_offset + 4;
 4
                                                                                                                              4
                                                                                                                                     always @(/*AUTOSENSE*/clipper_grant_count or service_clipper)
              end
 5
        end // always @ (posedge sclk)
                                                                                                                                      begin
                                                                                                                              6
                                                                                                                                          if(service clipper)
7
       wire pa_req_aux;
                                                                                                                                           clipper\_timeup = 1"b0;
8
       assign \quad \  pa\_req\_aux = q\_pa\_req\_control[1];
                                                                                                                                          else if(~clipper_grant_count[0] & ~clipper_grant_count[1])
                                                                                                                                           clipper_timeup = 1'b1;
10
       wire [1:0] pa_req_offset;
                                                                                                                             10
11
       assign pa_req_offset =q_pa_req_control[3:2];
                                                                                                                             11
                                                                                                                                           clipper_timeup = 1'b0;
12
                                                                                                                             12
13
                                                                                                                             13
       //0x40 is the bottom of the position export buffer
14
       //in a single (no aux buffer mode) there can be 4 buffers present in the position export buffer
                                                                                                                             14
                                                                                                                                     always @(/*AUTOSENSE*/rb0_grant_count or service_rb0)
15
                                                                                                                             15
       //at the following offsets : 0x40 , 0x44 , 0x48, 0x4c
        //otherwise only two buffers can fit in...at offsets 0x40 and 0x48 with their aux buffer at 0x44
                                                                                                                             16
                                                                                                                                          if(service_rb0)
                                                                                                                             17
                                                                                                                                           rb0_timeup = 1'b0;
18
                                                                                                                             18
                                                                                                                                          else if(~rb0_grant_count[0] & ~rb0_grant_count[1])
      assign pos_exp_read_ptr[6] = 1'b1; //located at offset 0x40...the upper 16 entries of the export buffer are dedicated to position data
                                                                                                                                           rb0_timeup = 1'b1;
                                                                                                                             20
                 pos_exp_read_ptr[5:4] = 2'b00; //only 16 entries dedicated to position...use only
21
22
                                                                                                                             21
                                                                                                                                           rb0 timeup = 1'b0;
      assign \quad pos\_exp\_read\_ptf[3:0] = (pa\_req\_aux) ? pos\_exp\_read\_offset + pa\_req\_offset + 4 \\ pos\_exp\_read\_offset + pa\_req\_offset;
                                                                                                                             22
                                                                                                                             23
25
                                                                                                                                     always @(/*AUTOSENSE*/rb1_grant_count or service_rb1)
                                                                                                                             24
26
       wire pa_req_sp_id;
                                                                                                                             25
       assign pa_req_sp_id = q_pa_req_control[4];
                                             Page 91 of 100
                                                                                                                                                                          Page 92 of 100
                                                                       Ex. 2106 - export control.v
                                                                                                                                                                                                    Ex. 2106 - export control.v
```

```
if(service rb1)
                                                                                                                                     rb3 timeup = 1'b0;
 2
             rb1 timeup = 1'b0;
                                                                                                                         2
            else if(\simrb1_grant_count[0] & \simrb1_grant_count[1])
                                                                                                                         3
             rb1_timeup = 1'b1;
                                                                                                                         4
                                                                                                                               //combinational logic for the above state machine
                                                                                                                               always @(/*AUTOSENSE*/'GRANT_CLIPPER or 'GRANT_RB0 or 'GRANT_RB1
              rb1_timeup = 1'b0;
                                                                                                                                       or `GRANT_RB2 or `GRANT_RB3 or `READ_IDLE
                                                                                                                                       or pos_exp_read_ptr or rb0_read_index or rb1_read_index
                                                                                                                                       or rb2_read_index or rb3_read_index or read_state)
       always @(/*AUTOSENSE*/rb2_grant_count or service_rb2)
                                                                                                                                begin
10
                                                                                                                        10
                                                                                                                                    case(read_state)
11
            if(service_rb2)
                                                                                                                        11
                                                                                                                                     `READ_IDLE:
12
             rb2_timeup = 1'b0;
                                                                                                                        12
                                                                                                                                       begin
13
            else if(~rb2_grant_count[0] & ~rb2_grant_count[1])
                                                                                                                        13
                                                                                                                                        service_rb0 = 1'b0;
14
             rb2_timeup = 1'b1;
                                                                                                                                        service_rb1 = 1'b0;
15
                                                                                                                                        service_rb2 = 1'b0;
                                                                                                                        15
                                                                                                                                        service rb3 = 1'b0:
16
              rb2 timeup = 1'b0:
                                                                                                                        16
17
                                                                                                                        17
                                                                                                                                        service clipper = 1'b0;
                                                                                                                                        exp_read_pointer = 9'b0;
18
                                                                                                                        18
19
       always @(/*AUTOSENSE*/rb3 grant count or service rb3)
                                                                                                                        19
                                                                                                                                        exp_buff_read_en = 1'b0;
20
                                                                                                                        20
                                                                                                                                        read valid clipp = 1'b0;
21
                                                                                                                                        read valid rb0 = 1'b0:
            if(service rb3)
                                                                                                                        21
22
              rb3 timeup = 1'b0;
                                                                                                                        22
                                                                                                                                        read_valid_rb1 = 1'b0;
23
             else if(\simrb3_grant_count[0] & \simrb3_grant_count[1])
                                                                                                                        23
                                                                                                                                        read valid rb2 = 1'b0;
24
              rb3_timeup = 1'b1;
                                                                                                                        24
                                                                                                                                        read_valid_rb3 = 1'b0;
25
                                                                                                                        25
                                           Page 93 of 100
                                                                                                                                                                    Page 94 of 100
                                                                    Ex. 2106 - export_control.v
                                                                                                                                                                                            Ex. 2106 - export_control.v
              GRANT_CLIPPER:
                                                                                                                                        read_valid_rb0 = 1'b1;
                                                                                                                                        read_valid_rb1 = 1'b0;
 2
                                                                                                                         2
              begin
                service_rb0 = 1'b0;
                                                                                                                                        read valid rb2 = 1'b0:
                service rb1 = 1'b0;
                                                                                                                                        read valid rb3 = 1'b0:
                service rb2 = 1'b0:
                service_rb3 = 1'b0;
                                                                                                                                      'GRANT RBI-
                service_clipper = 1'b1;
                exp_read_pointer = pos_exp_read_ptr; //clipper base address ???? ANDI
                                                                                                                                        service rb0 = 1'b0;
                exp_buff_read_en = 1'b1;
                                                                                                                                        service_rb1 = 1'b1;
10
                read_valid_clipp = 1'b1;
                                                                                                                        10
                                                                                                                                        service_rb2 = 1'b0;
11
                read_valid_rb0 = 1'b0;
                                                                                                                        11
                                                                                                                                        service_rb3 = 1'b0;
12
                read_valid_rb1 = 1'b0;
                                                                                                                        12
                                                                                                                                         service_clipper = 1'b0;
13
                read_valid_rb2 = 1'b0;
                                                                                                                                         exp_read_pointer = rb1_read_index;
14
                read_valid_rb3 = 1'b0;
                                                                                                                                        exp_buff_read_en = 1'b1;
15
                                                                                                                                        read_valid_clipp = 1'b0;
16
              `GRANT_RB0:
                                                                                                                                        read_valid_rb0 = 1'b0;
17
                                                                                                                                        read_valid_rb1 = 1'b1;
               begin
18
                service_rb0 = 1'b1;
                                                                                                                                        read_valid_rb2 = 1'b0;
19
                service_rb1 = 1'b0;
                                                                                                                                        read_valid_rb3 = 1'b0;
20
                service rb2 = 1'b0;
                                                                                                                        20
                                                                                                                                       end
21
                service_rb3 = 1'b0;
                                                                                                                                      `GRANT_RB2:
                                                                                                                        21
22
                                                                                                                        22
                service clipper = 1'b0:
                                                                                                                                      begin
23
                exp_read_pointer = rb0_read_index;
                                                                                                                        23
                                                                                                                                        service rb0 = 1'b0:
24
                exp buff read en = 1'b1;
                                                                                                                        24
                                                                                                                                        service rb1 = 1'b0:
                read_valid_clipp = 1'b0;
                                                                                                                        25
                                                                                                                                        service_rb2 = 1'b1;
                                           Page 95 of 100
                                                                                                                                                                   Page 96 of 100
                                                                    Ex. 2106 - export control.v
                                                                                                                                                                                            Ex. 2106 - export control.v
```

```
service_rb3 = 1'b0;
                                                                                                                                        default:
 2
                service_clipper = 1'b0;
                                                                                                                                         begin
                exp_read_pointer = rb2_read_index;
                                                                                                                                           service_rb0 = 1'b0;
                exp_buff_read_en = 1'b1;
                                                                                                                                           service_rb1 = 1'b0;
                read_valid_clipp = 1'b0;
                                                                                                                                           service_rb2 = 1'b0;
                read_valid_rb0 = 1'b0;
                                                                                                                                           service_rb3 = 1'b0;
                read_valid_rb1 = 1'b0;
                                                                                                                                           service_clipper = 1'b0;
                read_valid_rb2 = 1'b1;
                                                                                                                                           exp_read_pointer = 9'b0;
                read_valid_rb3 = 1'b0;
                                                                                                                                           exp_buff_read_en = 1'b0;
10
                                                                                                                                           read_valid_clipp = 1'b0;
11
              `GRANT_RB3:
                                                                                                                                           read_valid_rb0 = 1'b0;
12
                                                                                                                                           read_valid_rb1 = 1'b0;
               begin
13
                service_rb0 = 1'b0;
                                                                                                                                           read_valid_rb2 = 1'b0;
14
                service_rb1 = 1'b0;
                                                                                                                                           read_valid_rb3 = 1'b0;
15
                service_rb2 = 1'b0;
                                                                                                                          15
16
                service_rb3 = 1'b1;
                                                                                                                          16
                                                                                                                                       endcase // case(read state)
17
                service_clipper = 1'b0;
                                                                                                                          17
18
                exp_read_pointer = rb3_read_index;
                                                                                                                          18
19
                exp buff read en = 1'b1;
                                                                                                                          19
20
                read valid clipp = 1'b0;
                                                                                                                          20
                                                                                                                                  always@(posedge sclk)
21
                read_valid_rb0 = 1'b0;
                                                                                                                          21
                                                                                                                                   begin
22
                read_valid_rb1 = 1'b0;
                                                                                                                          22
                                                                                                                                       if(srst)
23
                read valid rb2 = 1'b0;
                                                                                                                          23
24
                read_valid_rb3 = 1'b1;
                                                                                                                          24
                                                                                                                                          osx\_rb0\_index\_rtr <= 1"b0;
25
                                                                                                                          25
                                                                                                                                          osx\_rb1\_index\_rtr <= 1"b0;
                                            Page 97 of 100
                                                                                                                                                                       Page 98 of 100
                                                                      Ex. 2106 - export_control.v
                                                                                                                                                                                                Ex. 2106 - export_control.v
               osx_rb2_index_rtr <= 1'b0;
 2
                osx rb3 index rtr <= 1'b0;
              end
            else
               osx_rb0_index_rtr <= sx_to_rb0_index_rtr;
               osx\_rb1\_index\_rtr \mathrel{<=} sx\_to\_rb1\_index\_rtr;
               osx\_rb2\_index\_rtr \mathrel{<=} sx\_to\_rb2\_index\_rtr;
                osx\_rb3\_index\_rtr <= sx\_to\_rb3\_index\_rtr;
10
              end // else: !if(srst)
                                                                                                                           10
11
         end // always@ (posedge sclk)
                                                                                                                          11
12
                                                                                                                          12
13
       assign sx_rb0_index_rtr = osx_rb0_index_rtr;
14
       assign sx_rb1_index_rtr = osx_rb1_index_rtr;
15
        assign sx_rb2_index_rtr = osx_rb2_index_rtr;
16
        assign sx_rb3_index_rtr = osx_rb3_index_rtr;
17
18
19
      endmodule // export_control
20
21
22
23
24
25
                                            Page 99 of 100
                                                                                                                                                                      Page 100 of 100
                                                                      Ex. 2106 - export_control.v
                                                                                                                                                                                                Ex. 2106 - export_control.v
```

```
1 //
                     -*- Mode: Verilog -*-
 2 // Filename : macc.v
                                                                                                                 2 //this bus represents the ALU instruction word comming from the Sequencer
    \slash\hspace{-0.4em} // Description \slash\hspace{-0.4em} : Verilog wrapper for the MACC unit which includes the rest of the ALU opcodes
                                                                                                                      //at different cycles withing the 4 cycle window, the bus content represents
                                                                                                                 4 //different informantion as shown below
 5 // Author
                 : Andi Skende
                                                                                                                5 //Instruction word. The argument select part of the
 6 // Created On : Mon Oct 8 15:54:00 2001
                                                                                                                 6 //instruction is send over through four cycles since
                                                                                                                 7 //we only need one instruction every four cycles
 8 // Last Modified On:
                                                                                                                8 // cycle 0:SRC A Select
 9 // Update Count : 0
                                                                                                                            SRC A Argument Modifier
10 // Status : Unknown, Use with caution!
                                                                                                                10 //
11
                                                                                                                11
                                                                                                                     // cycle 1:SRC B Select
12 module macc(/*AUTOARG*/
                                                                                                                12 // SRC B Argument Modifier
13 // Outputs
                                                                                                                13 // SRC B Swizzle
14 oResult, oScalarInput, oScalarOpcode, oExportDst,
                                                                                                                14 // cycle 2:SRC C Select
15 // Inputs
                                                                                                                15
                                                                                                                      // SRC C Argument Modifier
16
     iRegData, iConstantData, iScalarData, iInstruction, iInstStart,
                                                                                                                16
17
                                                                                                                17
                                                                                                                      // cycle 3:Vector Opcode
18
                                                                                                                18
                                                                                                                     // Scalar Opcode
                                                                                                                19
                                                                                                                20 // Scalar Clamp
21
                                                                                                                21
                                                                                                                      // Vector Write Mask
22
      //These inputs represent all the possible sources from where the MACC arguments
                                                                                                                22
                                                                                                                23
24
                                                                                                                24
      input [127:0] iRegData, iConstantData, iScalarData;
25
                                                                                                                      input [20:0] iInstruction;
                                         Page 1 of 30
                                                                                                                                                         Page 2 of 30
                                                                                                                                                                                       Ex. 2107 - macc.v
                                                                       Ex. 2107 - macc.v
      input [0:0] iInstStart;
                                                                                                                                   MULADD = 5'h0b,
                                                                                                                                   CNDE = 5'h0c
      input
                sclk;
                                                                                                                                    CNDGE = 5'h0d
                                                                                                                                    CNDGT = 5'h0e:
      output [127:0] oResult;
                                                                                                                      //registering four sets of data from four different instruction transfer cycles
      output [31:0] oScalarInput;
      output [5:0] oScalarOpcode;
                                                                                                                 8
                                                                                                                      reg [20:0] q_Instruction0, q_Instruction1, q_Instruction2, q_Instruction3;
      output [5:0] oExportDst; //represents the destination pointer for exports
11 //ALU opcodes declared as parameters
                                                                                                                11
                                                                                                                       wire [5:0] ScalarOpcode;
12 //this definition is subject to change as more
                                                                                                                12
                                                                                                                      wire [0:0] VectorClamp,ScalarClamp;
13 //opcodes are added. for the latest definition
                                                                                                                      wire [3:0] VectorWriteMask,ScalarWriteMask;
14 //please refer to Shader Pipe Spec: ALU instruction definition
                                                                                                                14
                                                                                                                15
15
      parameter [4:0] ADD = 5'h00,
                                                                                                                      wire [31:0] MaccResult, Scalar Result;
17
                  MAX = 5'h02.
                                                                                                                17
                                                                                                                      reg [31:0] q0_MaccResultClamp, q1_MaccResultClamp, q2_MaccResultClamp;
                  MIN = 5'h03,
                                                                                                                18
                                                                                                                      reg [31:0] ResultMin;
18
                                                                                                                      wire [31:0] ResultMaxMin;
                  SETGT = 5'h05,
                                                                                                                20
                                                                                                                      reg [31:0] ResultMax;
                  SETGE = 5'h06.
                                                                                                                21
                                                                                                                      reg [31:0] q0_ResultMaxMin, q1_ResultMaxMin, q2_ResultMaxMin, q3_ResultMaxMin;
21
                  FRACT = 5'h08,
                                                                                                                23
                  TRUNC = 5'h09.
24
                                                                                                                24
                                                                                                                      //represents the previous instruction vector result
                  FLOOR = 5'h0a,
                                                                                                                                                         Page 4 of 30
                                         Page 3 of 30
                                                                       Ex. 2107 - macc.v
                                                                                                                                                                                       Ex. 2107 - macc.v
```

ATI 2107 LG v. ATI IPR2015-00326

```
wire [127:0] VectorData;
                                                                                                                                                                                                                             reg [31:0]
OperandCMod;
                                                                                                                                                                                                                                                                 OperandA, OperandB, OperandC,OperandAMod, OperandBMod,
 2
 3
            reg [127:0] InputDataA,InputDataB,InputDataC;
            reg [127:0] InputData;
                                                                                                                                                                                                                                //this state machine controls the four cycle GPR read/write and source sampling sequence
                                                                                                                                                                                                                                //state machine related variables
            //select logic controls
                                                                                                                                                                                                                                 reg [1:0]
                                                                                                                                                                                                                                                                 alu_state, next_alu_state;
                                                                                                                                                                                                                                                                 Opcode = 2'b11;
            wire [0:0]
                                              SrcASel,SrcBSel,SrcCSel;
                                                                                                             //select Reg vs. Vector vs. Scalar
10
                                                                                                                                                                                                                                                                 SrcB = 2'b01;
                                                                                                                                                                                                                                parameter
11
                                              SrcANegate, SrcBNegate, SrcCNegate; // input argument modifiers
                                                                                                                                                                                                                                                                 SrcC = 2'b10;
            wire [0:0]
12
13
             wire [1:0]
                                                                                                                                                                                                                                                            ONE = 32'h3f800000;
                                                                                  SrcAGreenSwizzle,
                                                                                                                                      SrcABlueSwizzle,
                                              SrcARedSwizzle,
                                                                                                                                                                                                                                parameter
          SrcAAlphaSwizzle;
                                                                                                                                                                                                                                                                ZERO = 32'h00000000;
                                                                                                                                                                                                                   13
                                                                                                                                                                                                                                parameter
             wire [1:0]
14
15
                                              SrcBRedSwizzle,
                                                                                   SrcBGreenSwizzle,
                                                                                                                                       SrcBBlueSwizzle.
                                                                                                                                                                                                                                                            GT ONE EXP = 8'h7f;
          SrcBAlphaSwizzle;
                                                                                                                                                                                                                                parameter
          wire [1:0]
SrcCAlphaSwizzle;
                                                                                                                                                                                                                   15
                                              SrcCRedSwizzle,
                                                                                        SrcCGreenSwizzle,\\
                                                                                                                                       SrcCBlueSwizzle,
                                                                                                                                                                                                                   16
             reg [0:0]
                                              decode\_SrcA, decode\_SrcB, decode\_SrcC, decode\_Opcode;
                                                                                                                                                                                                                   17
                                                                                                                                                                                                                                //state machine implementation
20
            reg [31:0]
                                              Src A Alpha Bus, Src A Red Bus, Src A Green Bus, Src A Blue Bus; \\
                                                                                                                                                                                                                   18
                                                                                                                                                                                                                               //This state machine has four states
                                                                                                                                                                                                                   19
                                                                                                                                                                                                                                //InstStart : first or default state
21
             reg [31:0]
                                              SrcBAlphaBus, SrcBRedBus, SrcBGreenBus, SrcBBlueBus;
                                                                                                                                                                                                                   20
                                                                                                                                                                                                                                                        : SrcA is sampled, selected and swizzled on its way to the first level of latches
22
             reg [31:0]
                                              SrcCAlphaBus, SrcCRedBus, SrcCGreenBus, SrcCBlueBus;
                                                                                                                                                                                                                                                        : SrcB is sampled, selected and swizzled on its way to the first level of latches
          \label{eq:condition} \begin{array}{lll} reg~[31:0] & SrcAAlphaBusLatch1, & SrcARedBusLatch0, SrcAGreenBusLatch0, SrcAGreenBusLatch1, SrcAGlueBusLatch0, SrcAGlueBusLatch1, SrcABlueBusLatch0, SrcABlueBusLatch1; \\ \end{array}
23
24
                                                                                                                                                                                                                   22
                                                                                                                                                                                                                                                        : SrcC is sampled, selected and swizzled on its way to the first level of latches
          \label{eq:continuity} reg~[31:0] & SrcBGreenBusLatch0, ~SrcBGreenBusLatch1, ~SrcBBlueBusLatch1, ~SrcBAlphaBusLatch1, ~SrcBAlphaBusLat
25
26
                                                                                                                                                                                                                   23
                                                                                                                                                                                                                   24
          reg [31:0] SrcCBlueBusLatch0,
SrcCRedBusLatch1, SrcCAlphaBusLatch1;
                                              SrcCBlueBusLatch1, SrcCGreenBusLatch1,
                                                                              Page 5 of 30
                                                                                                                                                                                                                                                                                                  Page 6 of 30
                                                                                                                                       Ex. 2107 - macc.v
                                                                                                                                                                                                                                                                                                                                                            Ex. 2107 - macc.v
            //registering the iInstStart event
                                                                                                                                                                                                                                  end
 2
                                                                                                                                                                                                                     2
                                              instruct issued;
                                                                                                                                                                                                                     3
                                                                                                                                                                                                                                 always @(alu_state or iInstStart)
 4
                                                                                                                                                                                                                     4
                                                                                                                                                                                                                                  begin
            always @(posedge sclk)
                                                                                                                                                                                                                                          case(alu state)
              begin
                                                                                                                                                                                                                                            SrcA:
                     if(srst)
                                                                                                                                                                                                                                                if(iInstStart | instruct_issued)
                         instruct_issued <= 1'b0;
                                                                                                                                                                                                                                                      next_alu_state = SrcB;
10
                                                                                                                                                                                                                    10
11
                     else if(iInstStart)
                                                                                                                                                                                                                   11
                                                                                                                                                                                                                                                      next_alu_state = SrcA;
12
                                                                                                                                                                                                                   12
                          instruct\_issued <= 1"b1;
13
                                                                                                                                                                                                                    13
14
                                                                                                                                                                                                                                             next_alu_state = SrcC;
15
               end // always @ (posedge sclk)
16
                                                                                                                                                                                                                                             next_alu_state = Opcode;
17
                                                                                                                                                                                                                                            Opcode:
18
                                                                                                                                                                                                                                             next alu state = SrcA;
19
                                                                                                                                                                                                                                          endcase
                                                                                                                                                                                                                   20
20
            always @(posedge sclk)
                                                                                                                                                                                                                                  end // always @ (state)
21
                                                                                                                                                                                                                   21
              begin
22
                                                                                                                                                                                                                   22
                     if(srst)
23
                                                                                                                                                                                                                                 //the decode_* signals are used throughout the logic to mark the current state of the state
                       alu state <= SrcA:
                                                                                                                                                                                                                   24
24
                                                                                                                                                                                                                   25
                                                                                                                                                                                                                                //any cycle out of the 4 cycle window from one instruction start to another
25
                        alu state <= next alu state;
                                                                              Page 7 of 30
                                                                                                                                                                                                                                                                                                  Page 8 of 30
                                                                                                                                       Ex. 2107 - macc.v
                                                                                                                                                                                                                                                                                                                                                           Ex. 2107 - macc.v
```

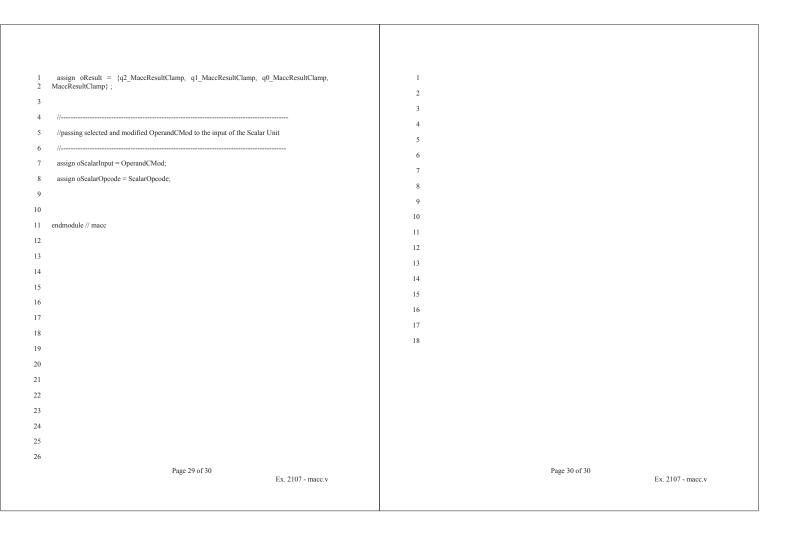
```
Opcode:
2
       always @(alu_state)
                                                                                                                                        begin
                                                                                                                                         decode_SrcA = 1'b0;
3
        begin
 4
            case (alu_state)
                                                                                                                                         decode_SrcB = 1'b0;
                                                                                                                                         decode_SrcC = 1'b0;
                                                                                                                                         decode_Opcode = 1'b1;
                decode_SrcA = 1'b1;
                decode_SrcB = 1'b0;
                decode_SrcC = 1'b0;
                decode_Opcode = 1'b0;
                                                                                                                         10
11
                                                                                                                         11
12
             SrcB:
                                                                                                                         12
13
                                                                                                                         13
              begin
14
               decode_SrcA = 1'b0;
                                                                                                                         14
                                                                                                                                //Registering the Instruction word (20 bits) in four consecutive cycles
15
               decode SrcB = 1'b1;
                                                                                                                         15
               decode SrcC = 1'b0:
16
                                                                                                                         16
                                                                                                                                 always@(posedge sclk)
17
                                                                                                                         17
               decode Opcode = 1'b0;
18
              end
                                                                                                                         18
                                                                                                                                   q_Instruction0 <= 21'b0;
19
             SrcC
                                                                                                                         19
                                                                                                                                  else if(decode SrcA)
                                                                                                                                   q_Instruction0 <= iInstruction;
20
                                                                                                                         20
21
               decode SrcA = 1'b0:
                                                                                                                         21
22
                decode_SrcB = 1'b0;
                                                                                                                         22
                                                                                                                                 always@(posedge sclk)
23
               decode_SrcC = 1'b1;
                                                                                                                         23
                                                                                                                                  q_Instruction1 <= 21'b0;
24
               decode_Opcode = 1'b0;
                                                                                                                         24
25
                                                                                                                                  else if(decode_SrcB)
                                             Page 9 of 30
                                                                                                                                                                      Page 10 of 30
                                                                              Ex. 2107 - macc.v
                                                                                                                                                                                                       Ex. 2107 - macc.v
         q_Instruction1 <= iInstruction;
2
                                                                                                                          2
                                                                                                                                assign SrcASel = q_Instruction0[2:0];
3
       always@(posedge sclk)
                                                                                                                                 assign\ SrcANegate = q\_Instruction0[3:3];
4
                                                                                                                                 assign\ Src AAlpha Swizzle = q\_Instruction 0[11:10];
        if(srst)
         q_Instruction2 <= 21'b0;
                                                                                                                                 assign\ SrcARedSwizzle = q\_Instruction0[5:4];
        else if(decode SrcC)
                                                                                                                                 assign\ Src A Green Swizzle = q\_Instruction 0 [7:6];
         q\_Instruction2 <= iInstruction;
                                                                                                                                 assign\ SrcABlueSwizzle = q\_Instruction0[9:8];
       always@(posedge sclk)
                                                                                                                                 assign\ SrcBSel = q\_Instruction1[2:0];
                                                                                                                                 assign\ SrcBNegate = q\_Instruction1[3:3];
10
                                                                                                                         10
11
         q_Instruction3 <= 21'b0;
                                                                                                                         11
                                                                                                                                 assign\ SrcBAlphaSwizzle = q\_Instruction1[11:10];
12
        else if(decode_Opcode)
                                                                                                                         12
                                                                                                                                 assign\ SrcBRedSwizzle = q\_Instruction1[5:4];
13
         q_Instruction3 <= iInstruction;
                                                                                                                                 assign SrcBGreenSwizzle = q_Instruction1[7:6]
14
                                                                                                                                 assign SrcBBlueSwizzle = q_Instruction1[9:8];
15
                                                                                                                         15
       //grabing the export destination ID.
16
       //If we are dealing with an export instruction...this value identifies which
                                                                                                                         16
17
      //attribute is being exported ...please refer to the shader pipe spec for more details
                                                                                                                         17
                                                                                                                                assign SrcCSel = q_Instruction2[2:0];
18
                                                                                                                         18
                                                                                                                                assign SrcCNegate = q Instruction2[3:3];
19
                                                                                                                                 assign SrcCAlphaSwizzle = q_Instruction2[11:10];
20
       assign oExportDst = q_Instruction0[17:12];
                                                                                                                         20
                                                                                                                                 assign SrcCRedSwizzle = q Instruction2[5:4];
21
                                                                                                                         21
                                                                                                                                 assign SrcCGreenSwizzle = q Instruction2[7:6]:
                                                                                                                         22
22
                                                                                                                                 assign\ SrcCBlueSwizzle = q\_Instruction2[9:8];
23
                                                                                                                         23
       //decoding the instruction word into a set of select/modify signals used
24
                                                                                                                         24
       //for argument selection and input modification on the way to MACC unit
25
                                                                                                                         25
                                            Page 11 of 30
                                                                                                                                                                      Page 12 of 30
                                                                             Ex. 2107 - macc.v
                                                                                                                                                                                                       Ex. 2107 - macc.v
```

```
assign VectorOpcode = q_Instruction3[4:0];
 2
       assign\ ScalarOpcode = q\_Instruction3[10:5];
                                                                                                                                 always@(SrcBSel or iConstantData or iRegData or VectorData or iScalarData)
       assign\ VectorClamp = q\_Instruction3[11:11];
                                                                                                                                 begin
       assign ScalarClamp = q_Instruction3[12:12];
                                                                                                                                     case(SrcBSel)
       assign VectorWriteMask = q_Instruction3[16:13];
                                                                                                                                      3'b000 : InputDataB = iConstantData;
       assign ScalarWriteMask = q_Instruction3[20:17];
                                                                                                                                      3'b100 : InputDataB = iRegData;
                                                                                                                                      3'b101 : InputDataB = iRegData;
                                                                                                                                      3'b110 : InputDataB = VectorData;
       //Argument Selectin for the three source operands going into the MACC unit
                                                                                                                                      3'b111 : InputDataB = iScalarData;
       //All information required for the selection logic in embedded into the ALU
                                                                                                                                      default: InputDataB = iRegData;
11
       //Instrution Word. Please refer to the Shade Processor Spec for a delailed
                                                                                                                                     endcase // case(SrcBSel)
12
       //definition of the select fields for the three sources
                                                                                                                                 end // always@ (SrcBSel or iConstantData or iRegData or iVectorData or iScalarData)
13
                                                                                                                         13
14
                                                                                                                         14
                                                                                                                                 always@(SrcCSel or iConstantData or iRegData or VectorData or iScalarData)
15
                                                                                                                         15
       always@(SrcASel or iConstantData or iRegData or VectorData or iScalarData)
                                                                                                                                 begin
16
                                                                                                                         16
                                                                                                                                     case(SrcCSel)
17
                                                                                                                         17
                                                                                                                                      3'b000 : InputDataC = iConstantData:
            case(SrcASel)
18
             3'b000 : InputDataA = iConstantData;
                                                                                                                         18
                                                                                                                                      3'b100 · InputDataC = iRegData:
19
             3'b100 : InputDataA = iRegData;
                                                                                                                         19
                                                                                                                                      3'b101 : InputDataC = iRegData;
20
             3'b101 : InputDataA = iRegData
                                                                                                                         20
                                                                                                                                      3'h110 : InnutDataC = VectorData:
21
             3'b110 : InputDataA = VectorData:
                                                                                                                         21
                                                                                                                                      3'b111 : InputDataC = iScalarData:
22
             3'b111 : InputDataA = iScalarData;
                                                                                                                                      default: InputDataC = iRegData;
23
             default: InputDataA = iRegData;
                                                                                                                         23
                                                                                                                                     endcase // case(SrcCSel)
            endcase // case(SrcASel)
24
                                                                                                                         24
                                                                                                                                  end // always@ (SrcCSel or iConstantData or iRegData or iVectorData or iScalarData)
        end // always@ (SrcASel or iConstantData or iRegData or iVectorData or iScalarData)
                                            Page 13 of 30
                                                                                                                                                                     Page 14 of 30
                                                                             Ex. 2107 - macc.v
                                                                                                                                                                                                      Ex. 2107 - macc.v
                                                                                                                                always@(InputDataA or SrcAGreenSwizzle)
       //Input Modifiers ie. swizzle and negate are begin applied
                                                                                                                                 case(SrcAGreenSwizzle)
                                                                                                                                  2'b00: SrcAGreenBus = InputDataA[63:32];
                                                                                                                                  2'b01: SrcAGreenBus = InputDataA[31:0];
       //Source A swizzling
                                                                                                                                   2'b10: SrcAGreenBus = InputDataA[127:96];
                                                                                                                                  2'b11: SrcAGreenBus = InputDataA[95:64];
       always@(InputDataA or SrcAAlphaSwizzle)
                                                                                                                                  endcase // case(SrcAGreenSwizzle)
10
        case(SrcAAlphaSwizzle) \\
                                                                                                                         10
11
         2'b00: SrcAAlphaBus = InputDataA[127:96];
                                                                                                                         11
                                                                                                                                always@(InputDataA or SrcABlueSwizzle)
12
         2'b01: SrcAAlphaBus = InputDataA[95:64];
                                                                                                                         12
                                                                                                                                 case(SrcABlueSwizzle)
13
          2'b10: SrcAAlphaBus = InputDataA[63:32];
                                                                                                                                  2'b00: SrcABlueBus = InputDataA[31:0];
14
         2'b11: SrcAAlphaBus = InputDataA[31:0];
                                                                                                                                  2'b01: SrcABlueBus = InputDataA[127:96];
15
        endcase // case(SrcAAlphaSwizzle)
                                                                                                                                  2'b10: SrcABlueBus = InputDataA[95:64];
16
                                                                                                                                  2'b11: SrcABlueBus = InputDataA[63:32];
17
                                                                                                                                 endcase // case(SrcAGreenSwizzle)
18
       always@(InputDataA or SrcARedSwizzle)
                                                                                                                         18
19
        case(SrcARedSwizzle)
                                                                                                                                //Source B swizzling
20
         2'b00: SrcARedBus = InputDataA[95:64];
                                                                                                                         20
21
         2'b01: SrcARedBus = InputDataA[63:32];
                                                                                                                         21
                                                                                                                                always@(InputDataB or SrcBAlphaSwizzle)
22
         2'b10: SrcARedBus = InputDataA[31:0]:
                                                                                                                         22
                                                                                                                                 case(SrcBAlphaSwizzle)
23
         2'b11: SrcARedBus = InputDataA[127:96]:
                                                                                                                         23
                                                                                                                                  2'b00: SrcBAlphaBus = InputDataB[127:96]:
24
        endcase // case(SrcARedSwizzle)
                                                                                                                         24
                                                                                                                                  2'b01: SrcBAlphaBus = InputDataB[95:64];
25
                                                                                                                                  2'b10: SrcBAlphaBus = InputDataB[63:32];
                                            Page 15 of 30
                                                                                                                                                                     Page 16 of 30
                                                                             Ex. 2107 - macc.v
                                                                                                                                                                                                      Ex. 2107 - macc.v
```

```
2'b11: SrcBAlphaBus = InputDataB[31:0];
                                                                                                                            2'b11: SrcBBlueBus = InputDataB[63:32];
2
        endcase
                                                                                                                           endcase // case(SrcBGreenSwizzle)
       always@(InputDataB or SrcBRedSwizzle)
                                                                                                                          //Source C swizzling
        case(SrcBRedSwizzle)
                                                                                                                           always@(InputDataC or SrcCAlphaSwizzle)
         2'b00: SrcBRedBus = InputDataB[95:64];
                                                                                                                           case(SrcCAlphaSwizzle)
         2'b01: SrcBRedBus = InputDataB[63:32];
                                                                                                                            2'b00: SrcCAlphaBus = InputDataC[127:96];
         2'b10: SrcBRedBus = InputDataB[31:0];
                                                                                                                            2'b01: SrcCAlphaBus = InputDataC[95:64];
         2'b11: SrcBRedBus = InputDataB[127:96];
                                                                                                                            2'b10: SrcCAlphaBus = InputDataC[63:32];
        endcase // case(SrcBRedSwizzle)
                                                                                                                            2"b11: SrcCAlphaBus = InputDataC[31:0];\\
11
                                                                                                                   11
12
       always@(InputDataB or SrcBGreenSwizzle)
                                                                                                                   12
13
                                                                                                                   13
                                                                                                                          always@(InputDataC or SrcCRedSwizzle)
        case(SrcBGreenSwizzle)
14
         2'b00: SrcBGreenBus = InputDataB[63:32];
                                                                                                                   14
                                                                                                                           case(SrcCRedSwizzle)
15
         2'b01: SrcBGreenBus = InputDataB[31:0];
                                                                                                                   15
                                                                                                                            2'b00: SrcCRedBus = InputDataC[95:64];
16
         2'b10: SrcBGreenBus = InputDataB[127:96]:
                                                                                                                   16
                                                                                                                            2'b01: SrcCRedBus = InputDataC[63:32]:
17
         2'b11: SrcBGreenBus = InputDataB[95:64]:
                                                                                                                   17
                                                                                                                            2'b10: SrcCRedBus = InputDataC[31:0]:
18
                                                                                                                            2'b11: SrcCRedBus = InputDataC[127:96];
        endcase // case(SrcBGreenSwizzle)
                                                                                                                   18
19
                                                                                                                   19
                                                                                                                           endcase // case(SrcCRedSwizzle)
20
                                                                                                                   20
21
       always@(InputDataB or SrcBBlueSwizzle)
                                                                                                                   21
                                                                                                                          always@(InputDataC or SrcCGreenSwizzle)
22
        case(SrcBBlueSwizzle)
                                                                                                                   22
                                                                                                                           case(SrcCGreenSwizzle)
23
         2'b00: SrcBBlueBus = InputDataB[31:0];
                                                                                                                   23
                                                                                                                            2'b00: SrcCGreenBus = InputDataC[63:32];
24
         2'b01: SrcBBlueBus = InputDataB[127:96];
                                                                                                                   24
                                                                                                                            2'b01: SrcCGreenBus = InputDataC[31:0];
25
         2'b10: SrcBBlueBus = InputDataB[95:64];
                                                                                                                            2'b10: SrcCGreenBus = InputDataC[127:96];
                                          Page 17 of 30
                                                                                                                                                             Page 18 of 30
                                                                          Ex. 2107 - macc.v
                                                                                                                                                                                             Ex. 2107 - macc.v
         2'b11: SrcCGreenBus = InputDataC[95:64];
                                                                                                                                 SrcAAlphaBusLatch1 <= SrcAAlphaBus;
2
                                                                                                                              SrcARedBusLatch0 <= SrcARedBus;
        endcase // case(SrcCGreenSwizzle)
                                                                                                                                 SrcAGreenBusLatch0 <= SrcAGreenBus:
                                                                                                                                 SrcABlueBusLatch0 <= SrcABlueBus:
       always@(InputDataC or SrcCBlueSwizzle)
        case(SrcCBlueSwizzle)
         2'b00: SrcCBlueBus = InputDataC[31:0];
                                                                                                                          always@(posedge sclk)
         2'b01: SrcCBlueBus = InputDataC[127:96];
                                                                                                                           if(decode_SrcC)
         2'b10: SrcCBlueBus = InputDataC[95:64];
         2'b11: SrcCBlueBus = InputDataC[63:32];
                                                                                                                                SrcBAlphaBusLatch1 <= SrcBAlphaBus;
10
                                                                                                                   10
11
        endcase // case(SrcCGreenSwizzle)
                                                                                                                   11
                                                                                                                                SrcBRedBusLatch1 <= SrcBRedBus;
12
                                                                                                                   12
                                                                                                                                SrcBGreenBusLatch0 <= SrcBGreenBus;
13
                                                                                                                   13
                                                                                                                                SrcBBlueBusLatch0 <= SrcBBlueBus;
14
                                                                                                                   14
15
                                                                                                                   15
16
      //Modeling stages for the Argument storing
                                                                                                                   16
17
                                                                                                                   17
                                                                                                                          always@(posedge sclk)
18
                                                                                                                   18
                                                                                                                           if(decode Opcode)
19
      // always@(SrcAAlphaBus or decode_SrcA)
                                                                                                                   19
                                                                                                                            begin
20
      // if(decode SrcA)
                                                                                                                   20
                                                                                                                                SrcCAlphaBusLatch1 <= SrcCAlphaBus;
21
      // SrcAAlphaBusLatch1 = SrcAAlphaBus;
                                                                                                                   21
                                                                                                                                SrcCRedBusLatch1 <= SrcCRedBus;
22
                                                                                                                   22
                                                                                                                                SrcCGreenBusLatch1 <= SrcCGreenBus:
23
                                                                                                                   23
                                                                                                                                SrcCBlueBusLatch0 <= SrcCBlueBus:
      always@(posedge sclk)
24
                                                                                                                   24
        if(decode SrcB)
25
                                                                                                                   25
                                          Page 19 of 30
                                                                                                                                                             Page 20 of 30
                                                                         Ex. 2107 - macc.v
                                                                                                                                                                                             Ex. 2107 - macc.v
```

```
//second level of latches
                                                                                                                                    end // always@ (sclk)
2
       always@(posedge sclk)
        if(decode SrcA)
                                                                                                                                    always@(posedge sclk)
              SrcARedBusLatch1 \le SrcARedBusLatch0;
              SrcAGreenBusLatch1 <= SrcAGreenBusLatch0;
                                                                                                                                         if(decode_SrcA)
              SrcABlueBusLatch1 <= SrcABlueBusLatch0; \\
                                                                                                                                          OperandB <= SrcBAlphaBusLatch1;
              SrcBGreenBusLatch1 <= SrcBGreenBusLatch0;
                                                                                                                                        else if(decode_SrcB)
              SrcBBlueBusLatch1 <= SrcBBlueBusLatch0;
                                                                                                                                         OperandB <= SrcBRedBusLatch1;
              SrcCBlueBusLatch1 \mathrel{<=} SrcCBlueBusLatch0;\\
                                                                                                                                        else if(decode_SrcC)
11
                                                                                                                                          OperandB <= SrcBGreenBusLatch1;
12
13
                                                                                                                            13
                                                                                                                                          OperandB <= SrcBBlueBusLatch1;
14
       // register the outputs from the latches into the MACC unit
                                                                                                                                    end // always@ (sclk)
15
                                                                                                                            15
16
                                                                                                                            16
       always@(posedge sclk)
                                                                                                                                    always@(posedge sclk)
17
                                                                                                                            17
        begin
                                                                                                                                    begin
18
            if(decode SrcA)
                                                                                                                            18
                                                                                                                                         if(decode SrcA)
19
             OperandA <= SrcAAlphaBusLatch1;
                                                                                                                            19
                                                                                                                                          OperandC <= SrcCAlphaBusLatch1;
20
            else if(decode SrcB)
                                                                                                                            20
                                                                                                                                         else if(decode SrcB)
21
             OperandA <= SrcARedBusLatch1:
                                                                                                                            21
                                                                                                                                         OperandC <= SrcCRedBusLatch1:
22
            else if(decode SrcC)
                                                                                                                            22
                                                                                                                                         else if(decode SrcC)
23
             OperandA <= SrcAGreenBusLatch1;
                                                                                                                            23
                                                                                                                                         OperandC <= SrcCGreenBusLatch1;
24
                                                                                                                            24
             OperandA <= SrcABlueBusLatch1;
                                                                                                                            25
                                                                                                                                          OperandC <= SrcCBlueBusLatch1;
                                             Page 21 of 30
                                                                                                                                                                         Page 22 of 30
                                                                               Ex. 2107 - macc.v
                                                                                                                                                                                                            Ex. 2107 - macc.v
        end // always@ (sclk)
                                                                                                                                    wire [1:0] opcode_mux_ctl;
2
                                                                                                                                  \label{eq:cond} \begin{array}{lll} \text{reg} & & & [1:0] \\ q0\_\text{opcode}\_\text{mux\_ctl}, q1\_\text{opcode}\_\text{mux\_ctl}, q2\_\text{opcode}\_\text{mux\_ctl}, q3\_\text{opcode}\_\text{mux\_ctl}, q4\_\text{opcode} \\ \_\text{mux\_ctl}; \end{array}
       //Input Modifier ....NEGATE.
5
                                                                                                                                   //generating control signals for routing the proper path into the final result
       always@(SrcANegate or OperandA)
        if(SrcANegate)
         Operand A Mod [31:0] = \{Operand A [31] \land Src A Negate, Operand A [30:0]\};
10
         Operand A Mod = Operand A; \\
                                                                                                                            11
                                                                                                                                   assign opcode_mux_ctl[0] = (VectorOpcode == MIN);
11
                                                                                                                                    assign opcode_mux_ctl[1] = (VectorOpcode == MAX) ;
12
       always@(SrcBNegate or OperandB)
                                                                                                                            13
13
                                                                                                                            14
14
         OperandBMod[31:0] = \{OperandB[31] \land SrcBNegate, OperandB[30:0]\};
                                                                                                                            15
                                                                                                                                    always @(posedge sclk)
15
                                                                                                                            16
                                                                                                                                    begin
16
         OperandBMod = OperandB;
                                                                                                                            17
                                                                                                                                         q0_opcode_mux_ctl <= opcode_mux_ctl;
17
                                                                                                                            18
                                                                                                                                        ql_opcode_mux_ctl <= q0_opcode_mux_ctl;
18
       always@(SrcCNegate or OperandC)
                                                                                                                            19
                                                                                                                                         q2_opcode_mux_ctl <= q1_opcode_mux_ctl;
19
                                                                                                                            20
                                                                                                                                         q3_opcode_mux_ctl <= q2_opcode_mux_ctl;
20
         OperandCMod[31:0] = \{OperandC[31] \land SrcCNegate, OperandC[30:0]\};
                                                                                                                           21
                                                                                                                                         q4\_opcode\_mux\_ctl \mathrel{<=} q3\_opcode\_mux\_ctl;
21
                                                                                                                           22
22
         OperandCMod = OperandC:
                                                                                                                           23
23
                                                                                                                            24
24
                                                                                                                                   //Floating point Multiply and Accumulate
25
                                             Page 23 of 30
                                                                                                                                                                         Page 24 of 30
                                                                               Ex. 2107 - macc.v
                                                                                                                                                                                                            Ex. 2107 - macc.v
```

```
macc32
                        mad(OperandAMod,
                                                     OperandBMod,
                                                                             OperandCMod
                                                                                                                                     end
      VectorOpcode,MaccResult,sclk);
                                                                                                                                   end \mathbin{/\!/} if (OperandAMod[30:0] \mathbin{>=} OperandBMod[30:0])
                                                                                                                                 else\ if\ (OperandBMod[30:0]>=OperandAMod[30:0])
                                                                                                                                    if(!OperandBMod[31])
       //some of the opcodes do not have to be implemented via the MACC unit
       //for example : MAX can be implemented via compares of the exponents and/or mantissas of
                                                                                                                                         ResultMax = OperandBMod;
       //the two numbers assuming that the numbers are normalized
                                                                                                                                         ResultMin = OperandAMod;
       //this is a separate parallel pipeline from the MACC
10
11
                                                                                                                                     begin
12
       //MIN or MAX
                                                                                                                                         ResultMax = OperandAMod;
13
       //revisit this logic for the case when exp = 0 ...ANDI
                                                                                                                                         ResultMin = OperandBMod;
       always @(/*AUTOSENSE*/OperandAMod or OperandBMod)
14
15
                                                                                                                     15
                                                                                                                                  end \mathbin{/\!/} if (OperandBMod[30:0] >= OperandAMod[30:0])
16
            if(OperandAMod[30:0] >= OperandBMod[30:0]) \\
                                                                                                                     16
                                                                                                                             end // always @ (...
17
                                                                                                                     17
18
              if(!OperandAMod[31])
                                                                                                                     18
19
                                                                                                                     19
                                                                                                                            //choose MIN vs MAX
20
                   ResultMax = OperandAMod; \\
                                                                                                                     20
                                                                                                                            assign\ ResultMaxMin = (opcode\_mux\_ctl[1])\ ?\ ResultMax: ResultMin;
21
                    ResultMin = OperandBMod;
                                                                                                                     21
22
                                                                                                                         //delay the ResultMaxMin to match with the other path of the pipeline that goes through the MACC
                                                                                                                     22
23
23
24
                begin
                                                                                                                     24
                                                                                                                            always@(posedge sclk)
25
                   ResultMax = OperandBMod;
                                                                                                                     25
                    ResultMin = OperandAMod;
                                                                                                                                 q0_ResultMaxMin <= ResultMaxMin;
                                          Page 25 of 30
                                                                                                                                                               Page 26 of 30
                                                                           Ex. 2107 - macc.v
                                                                                                                                                                                                Ex. 2107 - macc.v
                                                                                                                                 if(ResultClamp)
            q1 ResultMaxMin <= q0 ResultMaxMin;
            q2 ResultMaxMin <= q1 ResultMaxMin;
                                                                                                                                  begin
            q3 ResultMaxMin <= q2 ResultMaxMin;
                                                                                                                                    if(MaccResultMux[31])
                                                                                                                                     MaccResultClamp = ZERO;
                                                                                                                                    else if(MaccResultMux[30:24] > GT_ONE_EXP)
       reg [31:0] MaccResultMux;
                                                                                                                                   MaccResultClamp = ONE;
                                                                                                                                     MaccResultClamp = MaccResultMux;
       //Routing the Result into MaccResultMux based on the opcode
10
                                                                                                                     10
11
       always @(/*AUTOSENSE*/MaccResult or q3_ResultMaxMin
                                                                                                                     11
                                                                                                                                  MaccResultClamp = MaccResultMux; \\
12
                                                                                                                     12
                                                                                                                             end // always@ (MaccResult or ResultClamp)
13
                                                                                                                     13
14
                                                                                                                     14
            case(q4_opcode_mux_ctl)
15
            2'b00: MaccResultMux = MaccResult;
                                                                                                                            //pipeline delays for the code....creating the 4 stage delay for the VectorResult
16
             2'b01: MaccResultMux = q3_ResultMaxMin;
                                                                                                                     16
17
             2'b10: MaccResultMux = q3_ResultMaxMin;
                                                                                                                     17
                                                                                                                            always@(posedge sclk)
18
             default : MaccResultMux = MaccResult;
                                                                                                                     18
19
                                                                                                                     19
            endcase // case(opcode mux ctl)
                                                                                                                                 q0 MaccResultClamp <= MaccResultClamp;
                                                                                                                     20
20
        end
                                                                                                                                 q1 MaccResultClamp <= q0 MaccResultClamp;
21
                                                                                                                     21
                                                                                                                                 q2_MaccResultClamp <= q1_MaccResultClamp;
22
                                                                                                                     22
23
                                                                                                                     23
       //Clamping the result and other output modifiers
                                                                                                                          assign\ Vector Data = \{q2\_MaccResultClamp,\ q1\_MaccResultClamp,\ q0\_MaccResultClamp,\ MaccResultClamp\};
24
       always@(/*AUTOSENSE*/MaccResultMux or ResultClamp)
25
                                          Page 27 of 30
                                                                                                                                                                Page 28 of 30
                                                                           Ex. 2107 - macc.v
                                                                                                                                                                                                Ex. 2107 - macc.v
```



1	// -*- Mode: Verilog -*-	1 output unsigned [31:0] result;
	// Filename : macc32.mc	2
2		
4	// Description : This file represents the implementation of the MACC unit (Multiply and Accumulate)	- "
5	// Author : Andi Skende	4 // ALU opcode list declaration
6	// Created On : Mon Jan 28 16:04:47 2002	5 //
7	// Last Modified By: .	6 #define ADD 5'h00
8	// Last Modified On: .	7 #define MUL 5'h01
		8 #define MAX 5'h02
9	// Update Count : 0	9 #define MIN 5'h03
10	// Status : Unknown, Use with caution!	10 #define SETE 5'h04
11		11 #define SETGT 5'h05
12	module macc32(in1,in2,in3,opcode,result);	12 #define SETGE 5'h06
13	directive(delay = 2400,pipeline = "on");	13 #define SETNE 5'h07
14	directive(clock = "iSCLK");	14 #define FRACT 5'h08
15	directive(multtype="booth");	
16	directive(fatype = "fastcla");	15 #define TRUNC 5'h09
17		16 #define FLOOR 5'h0a
18		17 #define MULADD 5'h0b
19	//	18 #define CNDE 5'h0c
		19 #define CNDGE 5'h0d
20	//declaration of the input data as 32-bit IEEE floating point with an implicit 1 as bit 24 of the	20 #define CNDGT_5'h0e
21	//mantissa	21
22	//	22 //declaring a couple of constants
23	input unsigned [31:0] in1;	23 //
24	input unsigned [31:0] in2;	24 wire unsigned [0:0] one = 1'h1;
25	input unsigned [31:0] in3;	
26	input unsigned [4:0] opcode;	25 wire unsigned [0:0] zero = 1'h0;
	Page 1 of 8	Page 2 of 8
	Ex. 2108 - macc32.mc	Ex. 2108 - macc32.mc
1	wire unsigned [31-0] one iges = 31h38800000-	1 wire signed (24.0) mant1 signed = mant1 one
1	wire unsigned [31:0] one_ieee = 31'h3f800000;	1 wire signed [24:0] mantl_signed = -mantl_one; 2 wire signed [24:0] mantl_signed = -mantl_one;
2		2 wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one ;
2	//control signals related to opcode	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one;</pre>
2 3 4	//control signals related to opcode	wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one;
2	//control signals related to opcode // wire unsigned [0:0] opcode_add = (opcode == ADD) ? one :zero;	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one;</pre>
2 3 4	//control signals related to opcode	wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one;
2 3 4 5	//control signals related to opcode // wire unsigned [0:0] opcode_add = (opcode == ADD) ? one :zero;	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one;</pre>
2 3 4 5	//control signals related to opcode //	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one;</pre>
2 3 4 5 6 7	//control signals related to opcode //	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one;</pre>
2 3 4 5 6 7 8	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one; 8
2 3 4 5 6 7 8 9	//control signals related to opcode //	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one; </pre>
2 3 4 5 6 7 8 9	//control signals related to opcode //	<pre>wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one; // // // // // // // // // // // // //</pre>
2 3 4 5 6 7 8 9 10 11	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one; // // // // // // // // //
2 3 4 5 6 7 8 9 10 11 12	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one; // // // // // // // // //
2 3 4 5 6 7 8 9 10 11 12 13	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1) ? mant1_signed : mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2) ? mant2_signed : mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3) ? mant3_signed : mant3_one; // // // // // // // /
2 3 4 5 6 7 8 9 10 11 12 13 14	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 3 4 4 5 5 6 6 7 7 8 9 10 11 12 13 14 15 16	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 3 4 4 5 5 6 6 7 7 8 9 10 11 12 13 14 15 16	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 3 4 5 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 3 4 5 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 3 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //
2 2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // selection logic for all three paths exp, mantissa and sign based on the opcode // // // // // // // // // // // // //
2 3 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // selection logic for all three paths exp, mantissa and sign based on the opcode // // // // // // // // // // // // //
2 2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // selection logic for all three paths exp, mantissa and sign based on the opcode // // // // // // // // // // // // //
2 3 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // selection logic for all three paths exp, mantissa and sign based on the opcode // // // // // // // // // // // // //
2 3 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant2 = (sign2)? mant2_signed: mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; // wire unsigned [10:0] opcode for all three paths exp, mantissa and sign based on the opcode // wire unsigned [0:0] opcode_mul_add = opcode_add opcode_muladd opcode_mul; wire unsigned [0:0] opcode_mul_add = opcode_add opcode_muladd opcode_mul; wire unsigned [1:0] opcode_sel = cat(opcode_mul_add, opcode_frac_trunc); wire unsigned [0:0] opcode_sel = opcode_mul_add; // wire unsigned [0:0] opcode_mul_add opcode_mul_add; // wire un
2 3 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	//control signals related to opcode //	wire signed [24:0] mant1 = (sign1)? mant1_signed: mant1_one; wire signed [24:0] mant2_signed = -mant2_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3_signed = -mant3_one; wire signed [24:0] mant3 = (sign3)? mant3_signed: mant3_one; //

ATI 2108 LG v. ATI IPR2015-00326

```
wire signed [8:0] exp1_plus_exp2 = exp1 + exp2;
 2
       wire signed [8:0] exp_mul = (opcode_sel) ? exp1_plus_exp2 : exp1;
                                                                                                                                    // denormalizing of the data before addition
       wire signed [8:0] exp_mul_minus_exp3 = exp_mul - exp3;
                                                                                                                              3
       wire signed [8:0] exp_delta =(opcode_sel) ? exp_mul_minus_exp3 : exp_mul;
                                                                                                                                  wire signed [25:0] mant_in1_in2_shf_24 = (exp_delta_gt_24) ? 26'h0 : mant_in1_in2_shf >> exp_delta_abs;
                                                                                                                                   wire signed [25:0] mant_mul = (exp_delta > 127) ? mant_in1_in2_shf: (mant_in1_in2_shf: 24);
       //detection of whether the number (in1) is all-fraction value (number < 1)
       wire unsigned [0:0] fract_all_fract = (exp_delta < 127);
                                                                                                                                    wire signed [25:0] mant3_shft = (exp_delta_gt_24) ? 25'h0 : mant3 >> exp_delta_abs;
                                                                                                                                     wire signed [24:0] mant3_denorm = (exp_delta > 127) ? mant3_shft : mant3;
       wire signed [7:0] exp_delta_unbiased = exp_delta - 127;
       wire unsigned [7:0] exp_delta_unsigned = - exp_delta_unbiased;
                                                                                                                             11
      wire unsigned [7:0] exp_delta_abs = (exp_delta_unbiased > 0)? exp_delta_unbiased: exp_delta_unsigned;
11
12
                                                                                                                             12
                                                                                                                                    //addition of the product result with the third argument into the macc unit
                                                                                                                             13
      wire unsigned [0:0] exp_delta_gt_24 = (exp_delta_abs > 5'h18) ? one : zero ;//flag to avoid shifting out for more than 24 positions
                                                                                                                             14
                                                                                                                                     wire signed [26:0] mant res = mant mul + mant3 denorm;
15
                                                                                                                             15
16
       //special logic related to FRACT and TRUNC logic
                                                                                                                             16
17
       wire unsigned [23:0] mant_fract = mant1_one << exp_delta_abs;
                                                                                                                             17
                                                                                                                                     wire signed [26:0] mant_result = mant_res >> 1;
      18
                                                                                                                                     wire signed [8:0] exp_mul_minus126 = exp_mul - 126;
                                                                                                                             19
                                                                                                                                     //wire \hspace{0.2cm} signed \hspace{0.1cm} \texttt{[8:0]} \hspace{0.1cm} exp\_mul\_m126 = (opcode\_sel) \hspace{0.1cm} ? \hspace{0.1cm} exp\_mul\_minus126 : 9'h7f; \\
20
                                                                                                                             20
                                                                                                                                     wire signed [8:0] exp2_plus1 = exp2 + 1;
21
                                                                                                                                     wire signed [8:0] exp_res = (exp_delta >= 127) ? exp_mul_minus126 : exp2_plus1;
22
       //multiplying ....in1*in2
                                                                                                                             22
23
                                                                                                                                     wire signed [7:0] exp_fract = (fract_all_fract) ? exp1 : 8'h7f;
24
        wire signed [48:0] mant_in1_in2 = mant1 * mant2;
                                                                                                                                     wire signed [8:0] exp_result = (opcode_sel) ? exp_res : exp_fract;
      wire signed [25:0] mant_in1_in2_shf = mant_in1_in2 >> 23; //andi ...no need to shift...just mask out the bottom 23 bits
27
                                                                                                                                     wire unsigned [26:0] mant_result_neg = -mant_result;
                                               Page 5 of 8
                                                                                                                                                                            Page 6 of 8
                                                                            Ex. 2108 - macc32.mc
                                                                                                                                                                                                         Ex. 2108 - macc32.mc
        wire unsigned [26:0] mant res unsigned = (mant res > 0) ? mant result: mant result neg;
 2
 4
       // this mux can be larger with the other opcodes being added in time
        wire unsigned [23:0] mant_res_unormalized = (opcode_sel) ? mant_res_unsigned[23:0]:
       wire unsigned [7:0] exp_modify = LZ(mant_res_unormalized);
        wire unsigned [7:0] exp_final = exp_result - exp_modify;
10
11
        wire unsigned [0:0] sign_add_final = (mant_res > 0) ? 0:1;
12
        wire unsigned [0:0] sign_final = (opcode_sel) ? sign_add_final : sign1;
13
14
        wire unsigned [23:0] mant_final_shifted = mant_res_unormalized << exp_modify;
15
        wire unsigned [23:0] mant_final = mant_final_shifted;
16
17
        wire unsigned [31:0] result_temp = cat(sign_final, exp_final, mant_final[22:0]);
18
19
20
       //defining the delay as 4 pipeline stages
21
22
       result = ResolveLatency(result_temp,5);
23
       //result = result_temp;
24
25
      endmodule // macc32
                                               Page 7 of 8
                                                                                                                                                                            Page 8 of 8
                                                                            Ex. 2108 - macc32.mc
                                                                                                                                                                                                         Ex. 2108 - macc32.mc
```

```
1 //
                     -*- Mode: Verilog -*-
                                                                                                            1 SX PA data.
 2 // Filename : sx.v
                                                                                                            2 // Inputs
 3 // Description : Shader Export top level
                                                                                                                  CG\_SX\_pm\_enb, SC\_SX\_quad\_x, SC\_SX\_quad\_y, SC\_SX\_quad\_mask,
 4 // Author : Andi Skende
                                                                                                                 SC\_SX\_quad\_tilex, SC\_SX\_quad\_tiley, SC\_SX\_quad\_send, sclk\_global,
 5 // Created On : Thu Mar 21 13:59:48 2002
                                                                                                            5 srst. SO SX interp flat vtx. SO SX interp flat gouraud.
                                                                                                            6 SQ_SX_interp_cyl_wrap, SQ_SX_pc_ptr0, SQ_SX_pc_ptr1,
 7 // Last Modified On:
                                                                                                            7 SQ_SX_pc_ptr2, SQ_SX_rt_sel, SQ_SX_pc_wr_en, SQ_SX_pc_wr_addr,
 8 // Update Count : 0
                                                                                                            8 SQ SX pc channel mask, SP0 SX data0, SP0 SX data1, SP0 SX data2,
               : Unknown, Use with caution!
                                                                                                                   SP0\_SX\_data3, SP1\_SX\_data0, SP1\_SX\_data1, SP1\_SX\_data2,
                                                                                                            10 SP1_SX_data3, SP0_SX_exp_pvalid, SP1_SX_exp_pvalid,
11 'timescale 1ns / 1ps
                                                                                                           11 SP0 SX exp alu id, SP1 SX exp alu id, SP0 SX exporting,
12 module sx(/*AUTOARG*/
                                                                                                           12 SP1_SX_exporting, SP0_SX_exp_dest, SP1_SX_exp_dest,
13 // Outputs
                                                                                                           13 SQ_SX_exp_type, SQ_SX_exp_number, SQ_SX_exp_state, SQ_SX_exp_id,
14 SX SC quad rtr, SX RBB rs out, SX RBB rd out, SX RBBM busy,
                                                                                                           14 SQ SX exp valid, SQ SX free done, SQ SX free id,
                                                                                                           15
15
                                                                                                                 RBBM_SX_soft_reset, RBBM_we, RBBM_wd, RBBM_a, RBBM_be, RBBM_re,
     SX_pipe_we, SX_pipe_re, SX_pipe_a, SX_pipe_wd,
      SX\_SQ\_exp\_count\_rdy, SX\_SQ\_exp\_pos\_avail, SX\_SQ\_exp\_buf\_avail, \\
                                                                                                                  RBB\_rs\_in, RBB\_rd\_in, SX\_in\_vtx\_data0, SX\_in\_vtx\_data1,
17 SX_SP_vtx_data0, SX_SP_vtx_data1, SX_SP_vtx_data2,
                                                                                                           17 SX in vtx data2, RB0 SX quad rtr, RB1 SX quad rtr,
18 SX out vtx data0, SX out vtx data1, SX out vtx data2,
                                                                                                           18 RB2 SX quad rtr, RB3 SX quad rtr, RB0 SX color rtr,
19 SX_RB_quad_x, SX_RB_quad_y, SX_RB_quad_mask, SX_RB_quad_type,
                                                                                                           19 RB1_SX_color_rtr, RB2_SX_color_rtr, RB3_SX_color_rtr,
20 SX_RB_quad_pixel, SX_RB_quad_index, SX_RB0_quad_send,
                                                                                                           20 RB0_SX_index, RB1_SX_index, RB2_SX_index, RB3_SX_index,
     SX RB1 quad send, SX RB2 quad send, SX RB3 quad send,
                                                                                                           21 RB0 SX index send, RB1 SX index send, RB2 SX index send,
      SX_RB0_color_data, SX_RB1_color_data, SX_RB2_color_data,
                                                                                                                  RB3_SX_index_send, RB0_SX_index_op, RB1_SX_index_op,
      SX_RB3_color_data, SX_RB0_color_send, SX_RB1_color_send,
                                                                                                           23 RB2_SX_index_op, RB3_SX_index_op, PA_SX_req, PA_SX_sp_id,
                                                                                                           24 PA_SX_offset, PA_SX_aux, PA_SX_last
24 SX RB2 color send, SX RB3 color send, SX RB0 index rtr,
25 SX_RB1_index_rtr, SX_RB2_index_rtr, SX_RB3_index_rtr, SX_PA_send,
                                       Page 1 of 25
                                                                                                                                                   Page 2 of 25
                                                                       Ex. 2109 - sx.v
                                                                                                                                                                                   Ex. 2109 - sx.v
                                                                                                             1 wire [0:0]
                                                                                                                                    sx_quad_rtr, q_sx_sc_quad_rtr;
                                                                                                            2
                                                                                                                  ati dff_out #(1) usx_sc_quad_rtr(sclk,sx_quad_rtr,q_sx_sc_quad_rtr);
      parameter unit_id = 1'b0;
                                                                                                            4
                                                                                                                                SX_SC_quad_rtr = q_sx_sc_quad_rtr;
                                                                                                                  wire [1:0]
                                                                                                                                    q_sc_sx_quad_x;
     //Power managment control interface
                                                                                                                  wire [1:0]
                                                                                                                                    q_sc_sx_quad_y;
     input [0:0] CG_SX_pm_enb;
                                                                                                                  wire [31:0]
                                                                                                                                    q_sc_sx_quad_mask;
                                                                                                                   wire [1:0]
                                                                                                                                    q sc sx quad tilex;
                                                                                                                   wire [0:0]
11
      //SC to SX Quad info interface
                                                                                                            11
                                                                                                                  wire [0:0]
                                                                                                                                     q_sc_sx_quad_send;
                                                                                                           12
12
      input [1:0]
                     SC_SX_quad_x;
                                                                                                            13 ati_dff_in #(2) usc_sx_quad_x(sclk,SC_SX_quad_x,q_sc_sx_quad_x);
14
      input [1:0]
                        SC_SX_quad_y;
                                                                                                           14 ati_dff_in #(2) usc_sx_quad_y(sclk,SC_SX_quad_y,q_sc_sx_quad_y);
15
      input [31:0] SC SX quad mask;
                                                                                                           15
                                                                                                                  ati dff in #(32) usc sx quad mask(sclk,SC SX quad mask,q sc sx quad mask);
                        SC_SX_quad_tilex;
                                                                                                                  ati\_dff\_in\ \#(2)\ usc\_sx\_quad\_tilex(sclk,SC\_SX\_quad\_tilex,q\_sc\_sx\_quad\_tilex);
17
                        SC SX quad tiley;
                                                                                                           17
                                                                                                                 ati\_dff\_in\ \#(1)\ usc\_sx\_quad\_tiley(sclk,SC\_SX\_quad\_tiley,q\_sc\_sx\_quad\_tiley);
                       SC SX quad send;
                                                                                                           18
18
      input
                                                                                                                 ati dff in #(1) usc sx quad send(sclk, SC SX quad send, q sc sx quad send);
      output
                       SX_SC_quad_rtr;
20
      input
                        sclk_global;
                                                                                                           20 //--
                                                                                                           21
                                                                                                                  //SO to SX Interpolation Bus/Parameter Cache-----
21
      input
                    srst:
23
      wire
                                                                                                                  input [1:0]
                                                                                                                                SQ_SX_interp_flat_vtx;
24
      assign
                    sclk = sclk_global;
                                                                                                                  input [0:0]
                                                                                                                                    SQ SX interp flat gouraud;
                                                                                                                  input [3:0]
                                                                                                                                     SQ_SX_interp_cyl_wrap;
                                        Page 3 of 25
                                                                                                                                                   Page 4 of 25
                                                                       Ex. 2109 - sx.v
                                                                                                                                                                                   Ex. 2109 - sx.v
```

ATI 2109 LG v. ATI IPR2015-00326

```
input [10:0] SQ_SX_pc_ptr0, SQ_SX_pc_ptr1, SQ_SX_pc_ptr2;
                                                                                                                             2
 2
       input [0:0]
                            SQ SX rt sel;
                                                                                                                                   input [3:0] SP0_SX_exp_pvalid, SP1_SX_exp_pvalid; //pixel valid mask
        input [0:0]
                            SQ_SX_pc_wr_en;
                                                                                                                                                SP0_SX_exp_alu_id, SP1_SX_exp_alu_id; //isn't one of these signals
        input [6:0]
                            SQ_SX_pc_wr_addr;
                                                                                                                                  input [0:0] SP0_SX_exporting, SP1_SX_exporting, //isn't one of these signals redundand \ref{eq:sparse} ANDI
        input [3:0]
                            SQ_SX_pc_channel_mask;
                                                                                                                                  input [5:0] SP0_SX_exp_dest, SP1_SX_exp_dest; //these are coming straight from the destination pointer of the ALU instruction
                                                                                                                                                               //SP does nothing else other than pipelining them through
       //Export Data Interface coming from SP going into Parameter Cache or Export Buffers
                                                                                                                            11
        input~[127:0]~SP0\_SX\_data0,SP0\_SX\_data1,SP0\_SX\_data2,SP0\_SX\_data3;
                                                                                                                            12
                                                                                                                                   wire [3:0] q_sp0_sx_exp_pvalid, q_sp1_sx_exp_pvalid;
11
        input~[127:0]~SP1\_SX\_data0,SP1\_SX\_data1,SP1\_SX\_data2,SP1\_SX\_data3;
                                                                                                                            13
                                                                                                                                   wire \qquad q\_sp0\_sx\_exp\_alu\_id, q\_sp1\_sx\_exp\_alu\_id;
12
                                                                                                                            14
                                                                                                                                   wire [0:0] q_sp0_sx_exporting, q_sp1_sx_exporting;
13
       wire [127:0] q sp0 sx data0, q sp0 sx data1, q sp0 sx data2, q sp0 sx data3;
                                                                                                                                   wire [5:0] q_sp0_sx_exp_dest, q_sp1_sx_exp_dest;
14
        wire [127:0] q_sp1_sx_data0, q_sp1_sx_data1, q_sp1_sx_data2, q_sp1_sx_data3;
15
                                                                                                                            17
       ati dff in #(128) usp0 sx data0(sclk,SP0 SX data0,q sp0 sx data0);
16
                                                                                                                            18
                                                                                                                                   ati\_dff\_in~\#(4)~usp0\_sx\_exp\_pvalid(sclk,SP0\_SX\_exp\_pvalid,q\_sp0\_sx\_exp\_pvalid);\\
17
       ati dff in #(128) usp0 sx data1(sclk,SP0 SX data1,q sp0 sx data1);
                                                                                                                            19
                                                                                                                                   ati\_dff\_in~\#(4)~usp1\_sx\_exp\_pvalid(sclk,SP1\_SX\_exp\_pvalid,q\_sp1\_sx\_exp\_pvalid);\\
18
        ati dff in #(128) usp0 sx data2(sclk,SP0 SX data2,q sp0 sx data2);
                                                                                                                                   ati\_dff\_in \ \#(1) \ usp0\_sx\_exp\_alu\_id(sclk,SP0\_SX\_exp\_alu\_id,q\_sp0\_sx\_exp\_alu\_id);
19
        ati dff in #(128) usp0 sx data3(sclk,SP0 SX data3,q sp0 sx data3);
                                                                                                                                    ati\_dff\_in\ \#(1)\ usp1\_sx\_exp\_alu\_id(sclk,SP1\_SX\_exp\_alu\_id,q\_sp1\_sx\_exp\_alu\_id);
20
                                                                                                                                   ati\_dff\_in~\#(1)~usp0\_sx\_exporting(sclk,SP0\_SX\_exporting,q\_sp0\_sx\_exporting);\\
21
       ati dff in #(128) usp1 sx data0(sclk,SP1 SX data0,q sp1 sx data0);
                                                                                                                                    ati_dff_in #(1) usp1_sx_exporting(sclk,SP1_SX_exporting,q_sp1_sx_exporting);
22
        ati\_dff\_in\ \#(128)\ usp1\_sx\_data1(sclk,SP1\_SX\_data1,q\_sp1\_sx\_data1);
                                                                                                                                   ati\_dff\_in\ \#(6)\ usp0\_sx\_exp\_dst(sclk,SP0\_SX\_exp\_dest,q\_sp0\_sx\_exp\_dest);
23
        ati\_dff\_in\ \#(128)\ usp1\_sx\_data2(sclk,SP1\_SX\_data2,q\_sp1\_sx\_data2);
                                                                                                                                   ati_dff_in #(6) usp1_sx_exp_dst(sclk,SP1_SX_exp_dest,q_sp1_sx_exp_dest);
24
        ati_dff_in #(128) usp1_sx_data3(sclk,SP1_SX_data3,q_sp1_sx_data3);
25
                                              Page 5 of 25
                                                                                                                                                                          Page 6 of 25
                                                                                  Ex. 2109 - sx.v
                                                                                                                                                                                                              Ex. 2109 - sx.v
                                                                                                                                   ati_dff_in #(1) usq_sx_free_id(sclk,SQ_SX_free_id,q_sq_sx_free_id);
       //SQ to SX Export Control Bus
       input [1:0] SQ SX exp type;
                                                                                                                                   //CP/RBBM Interface for Real Time data and snooping state registers
       input [1:0] SQ_SX_exp_number;
       input [2:0] SQ_SX_exp_state;
                                                                                                                                   //There's no rtr nrtrtr signals ...tie them high or low at the top level
       input [0:0] SQ SX exp id;
       input [0:0] SQ SX exp valid;
                                                                                                                                                RBBM SX soft reset:
        input [0:0] SQ_SX_free_done;
                                                                                                                                                RBBM we;
        input [0:0] SQ_SX_free_id;
10
                                                                                                                            10
                                                                                                                                   input [31:0] RBBM_wd;
11
                                                                                                                                   input [14:0] RBBM_a;
       wire [1:0] q_sq_sx_exp_type;
12
                                                                                                                                   input [3:0] RBBM_be;
                                                                                                                                               RBBM_re;
       wire [1:0] q_sq_sx_exp_number;
       wire [2:0] q_sq_sx_exp_state;
                                                                                                                                   input RBB rs in;
       wire [0:0] q_sq_sx_exp_id;
                                                                                                                                   input [31:0] RBB_rd_in;
16
       wire [0:0] q_sq_sx_exp_valid;
17
                                                                                                                            17
                                                                                                                                  wire q_rbbm_sx_soft_reset;
       wire [0:0] q_sq_sx_free_done;
18
       wire [0:0] q sq sx free id;
                                                                                                                                  wire q rbbm we;
                                                                                                                                   wire [31:0] q_rbbm_wd;
                                                                                                                                   wire [14:0] q rbbm a;
20
       ati_dff_in #(2) usq_sx_exp_type(sclk,SQ_SX_exp_type,q_sq_sx_exp_type);
                                                                                                                            20
21
                                                                                                                                   wire [3:0] q rbbm be;
       ati dff in #(2) usq sx exp number(sclk,SQ SX exp number,q sq sx exp number);
                                                                                                                            21
22
                                                                                                                            22
                                                                                                                                   wire a rbbm re:
       ati_dff_in #(3) usq_sx_exp_state(sclk,SQ_SX_exp_state,q_sq_sx_exp_state);
23
       ati\_dff\_in\ \#(1)\ usq\_sx\_exp\_id(sclk,SQ\_SX\_exp\_id,q\_sq\_sx\_exp\_id);
                                                                                                                            23
                                                                                                                                   wire q rbb rs in;
                                                                                                                                   wire [31:0] q_rbb_rd_in;
24
       ati_dff_in #(1) usq_sx_exp_valid(sclk,SQ_SX_exp_valid,q_sq_sx_exp_valid);
                                                                                                                            24
       ati\_dff\_in\ \#(1)\ usq\_sx\_free\_done(sclk,SQ\_SX\_free\_done,q\_sq\_sx\_free\_done);
                                              Page 7 of 25
                                                                                                                                                                          Page 8 of 25
                                                                                  Ex. 2109 - sx.v
                                                                                                                                                                                                              Ex. 2109 - sx.v
```



```
output [0:0] SX RB quad type;
                                                                                                                                 ati_dff_out #(32) usx_rb_quad_mask(sclk, sx_rb_quad_mask,q_sx_rb_quad_mask);
 2
       output [3:0] SX RB quad pixel;
                                                                                                                                  ati\_dff\_out \# (1) \ usx\_rb\_quad\_type (sclk, sx\_rb\_quad\_type, q\_sx\_rb\_quad\_type);
        output [7:0] SX RB quad index;
                                                                                                                                  ati\_dff\_out \# (4) \ usx\_rb\_quad\_pixel (sclk, sx\_rb\_quad\_pixel, q\_sx\_rb\_quad\_pixel);
     output [0:0]
SX_RB3_quad_send;
                            SX_RB0_quad_send, SX_RB1_quad_send, SX_RB2_quad_send,
                                                                                                                                  ati\_dff\_out \# (8) \ usx\_rb\_quad\_index(sclk, sx\_rb\_quad\_index, q\_sx\_rb\_quad\_index);
                                                                                                                                  ati_dff_out #(1) usx_rb0_quad_send(sclk, sx_rb0_quad_send,q_sx_rb0_quad_send);
                                                                                                                                  ati_dff_out #(1) usx_rb1_quad_send(sclk, sx_rb1_quad_send,q_sx_rb1_quad_send);
        wire [1:0] q_sx_rb_quad_x;
                                                                                                                                  ati\_dff\_out \# (1) \ usx\_rb2\_quad\_send(sclk, sx\_rb2\_quad\_send, q\_sx\_rb2\_quad\_send); \\
       wire [1:0] q_sx_rb_quad_y;
                                                                                                                                  ati\_dff\_out \#(1) \ usx\_rb3\_quad\_send(sclk, sx\_rb3\_quad\_send, q\_sx\_rb3\_quad\_send);
        wire [31:0] q_sx_rb_quad_mask;
10
        wire [0:0] q_sx_rb_quad_type;
                                                                                                                          10
                                                                                                                                              SX\_RB\_quad\_x = q\_sx\_rb\_quad\_x;
                                                                                                                                 assign
11
        wire [3:0] q sx rb quad pixel;
                                                                                                                          11
                                                                                                                                          SX_RB_quad_y = q_sx_rb_quad_y;
                                                                                                                                 assign
12
        wire [7:0] q sx rb quad index;
     12
                                                                                                                                           SX_RB_quad_mask = q_sx_rb_quad_mask;
                                                                                                                                 assign
                                                                                                                          13
                                                                                                                                           SX RB quad type = q sx rb quad type;
                                                                                                                                 assign
                                                                                                                          14
                                                                                                                                            SX RB quad pixel = q sx rb quad pixel;
15
                                                                                                                                  assign
                                                                                                                          15
                                                                                                                                           SX RB quad index = q sx rb quad index;
16
       wire [1:0] sx_rb_quad_x;
                                                                                                                                  assign
17
       wire [1:0] sx_rb_quad_y;
                                                                                                                          16
                                                                                                                                  assign
                                                                                                                                            SX RB0 quad send = q sx rb0 quad send;
                                                                                                                          17
18
        wire [31:0] sx_rb_quad_mask;
                                                                                                                                  assign
                                                                                                                                            SX RB1 quad send = q sx rb1 quad send;
19
        wire [0:0] sx_rb_quad_type;
                                                                                                                          18
                                                                                                                                  assign
                                                                                                                                            SX RB2 quad send = q sx rb2 quad send;
20
        wire [3:0] sx_rb_quad_pixel;
                                                                                                                          19
                                                                                                                                  assign
                                                                                                                                            SX_RB3_quad_send = q_sx_rb3_quad_send;
21
        wire [7:0] sx rb quad index;
                                                                                                                          20
22
                                                                                                                          21
        wire [0:0] sx_rb0_quad_send, sx_rb1_quad_send, sx_rb2_quad_send, sx_rb3_quad_send;
                                                                                                                          22
                                                                                                                                  input \ [0:0] \qquad RB0\_SX\_quad\_rtr, RB1\_SX\_quad\_rtr, RB2\_SX\_quad\_rtr, RB3\_SX\_quad\_rtr;
23
24
                                                                                                                          23
                                                                                                                                  wire [0:0]
                                                                                                                                               \verb|q_rb0_sx_quad_rtr,q_rb1_sx_quad_rtr,q_rb2_sx_quad_rtr,q_rb3_sx_quad_rtr|
25
        ati_dff_out #(2) usx_rb_quad_x(sclk, sx_rb_quad_x,q_sx_rb_quad_x);
                                                                                                                          24
                                                                                                                                  ati_dff_in #(1) urb0_sx_quad_rtr(sclk,RB0_SX_quad_rtr,q_rb0_sx_quad_rtr);
        ati_dff_out #(2) usx_rb_quad_y(sclk, sx_rb_quad_y,q_sx_rb_quad_y);
                                                                                                                                                                       Page 14 of 25
                                             Page 13 of 25
                                                                                 Ex. 2109 - sx.v
                                                                                                                                                                                                           Ex. 2109 - sx.v
        ati_dff_in #(1) urb1_sx_quad_rtr(sclk,RB1_SX_quad_rtr,q_rb1_sx_quad_rtr);
                                                                                                                                  ati dff out #(128) usx rb2 color data(sclk, sx rb2 color data,q sx rb2 color data);
 2
        ati dff in #(1) urb2 sx quad rtr(sclk,RB2 SX quad rtr,q rb2 sx quad rtr);
                                                                                                                                  ati dff out #(128) usx rb3 color data(sclk, sx rb3 color data,q sx rb3 color data);
        ati dff in #(1) urb3 sx quad rtr(sclk,RB3 SX quad rtr,q rb3 sx quad rtr);
                                                                                                                                  ati dff out #(1) usx rb0 color send(sclk, sx rb0 color send,q sx rb0 color send);
       assign
                    SX RB quad pixel = 4'b0;
                                                                                                                                  ati dff out #(1) usx rb1 color send(sclk, sx rb1 color send,q sx rb1 color send):
        assign
                    SX RB quad type = 1'b0;
                                                                                                                                  ati_dff_out #(1) usx_rb2_color_send(sclk, sx_rb2_color_send,q_sx_rb2_color_send);
                                                                                                                                  ati\_dff\_out \# (1) \ usx\_rb3\_color\_send (sclk, \ sx\_rb3\_color\_send, q\_sx\_rb3\_color\_send); \\
                                                                                                                                          SX_RB0_color_data = q_sx_rb0_color_data;
                                                                                                                                           SX_RB1_color_data = q_sx_rb1_color_data;
10
       //--Pixel Color Data interface-----
                                                                                                                          10
11
                                                                                                                          11
                                                                                                                                            SX_RB2_color_data = q_sx_rb2_color_data;
                                                                                                                          12
                                                                                                                                            SX_RB3_color_data = q_sx_rb3_color_data;
      SX_RB0_color_data,SX_RB1_color_data,SX_RB2_color_data,SX_RB3_color_data;
                                                                                                                          13
14
15
      output\\ SX\_RB0\_color\_send, SX\_RB1\_color\_send, SX\_RB2\_color\_send, SX\_RB3\_color\_send;
                                                                                                                          14
                                                                                                                                              SX_RB0_color_send = q_sx_rb0_color_send;
                                                                                                                                 assign
                                                                                                                          15
                                                                                                                                              SX_RB1_color_send = q_sx_rb1_color_send;
16
                                                                                                                                  assign
                                                                                                                          16
                                                                                                                                              SX_RB2_color_send = q_sx_rb2_color_send;
17
                                                                                                                                  assign
                                                                                                                          17
                                                                                                                                              SX_RB3_color_send = q_sx_rb3_color_send;
                                                                                                                                  assign
18
       wire [127:0] sx rb0 color data,sx rb1 color data,sx rb2 color data,sx rb3 color data;
                                                                                                                          18
19
        wire [0:0] sx_rb0_color_send,sx_rb1_color_send,sx_rb2_color_send,sx_rb3_color_send;
20
                                                                                                                                  input [0:0] RB0 SX color rtr,RB1 SX color rtr,RB2 SX color rtr,RB3 SX color rtr;
                                                                                                                          20
21
22
       q\_sx\_rb0\_color\_data, q\_sx\_rb1\_color\_data, q\_sx\_rb2\_color\_data, q\_sx\_rb3\_color\_data; 
                                                                                                                          21
                                                                                                                                  wire [0:0] q rb0 sx color rtr,q rb1 sx color rtr,q rb2 sx color rtr,q rb3 sx color rtr;
23
24
                                                                                                                          22
      q_sx_rb0_color_send,q_sx_rb1_color_send,q_sx_rb2_color_send,q_sx_rb3_color_send;
                                                                                                                          23
                                                                                                                                  ati dff in #(1) urb0 sx color rtr(sclk.RB0 SX color rtr.q rb0 sx color rtr):
25
                                                                                                                          24
                                                                                                                                  ati dff in #(1) urb1 sx color rtr(sclk,RB1 SX color rtr,q rb1 sx color rtr);
        ati\_dff\_out \# (128) \ usx\_rb0\_color\_data(sclk, sx\_rb0\_color\_data, q\_sx\_rb0\_color\_data);
                                                                                                                          25
                                                                                                                                  ati\_dff\_in\ \#(1)\ urb2\_sx\_color\_rtr(sclk,RB2\_SX\_color\_rtr,q\_rb2\_sx\_color\_rtr);
        ati_dff_out #(128) usx_rb1_color_data(sclk, sx_rb1_color_data,q_sx_rb1_color_data);
                                             Page 15 of 25
                                                                                                                                                                       Page 16 of 25
                                                                                 Ex. 2109 - sx.v
                                                                                                                                                                                                           Ex. 2109 - sx.v
```

```
ati dff in #(1) urb3 sx color rtr(sclk,RB3 SX color rtr,q rb3 sx color rtr);
  2
                                                                                                                                                                                                                                                                                             ati dff in #(8) urb0 sx index(sclk, RB0 SX index,q rb0 sx index);
                                                                                                                                                                                                                                                                                             ati dff in #(8) urb1 sx index(sclk, RB1 SX index,q rb1 sx index);
                                                                                                                                                                                                                                                                                             ati\_dff\_in\ \#(8)\ urb2\_sx\_index(sclk,\ RB2\_SX\_index,q\_rb2\_sx\_index);
               //SX to RB quad index interface---
                                                                                                                                                                                                                                                                                             ati_dff_in #(8) urb3_sx_index(sclk, RB3_SX_index,q_rb3_sx_index);
                                                                                                                                                                                                                                                                                             ati\_dff\_in\ \#(1)\ urb0\_sx\_index\_op(sclk, RB0\_SX\_index\_op,q\_rb0\_sx\_index\_op);
                input~[7:0] \qquad RB0\_SX\_index, RB1\_SX\_index, RB2\_SX\_index, RB3\_SX\_index;
             input \ [0:0] \\ RB0\_SX\_index\_send,RB1\_SX\_index\_send,RB2\_SX\_index\_send,RB3\_SX\_index\_send; \\
                                                                                                                                                                                                                                                                                             ati\_dff\_in\ \#(1)\ urb1\_sx\_index\_op(sclk, RB1\_SX\_index\_op,q\_rb1\_sx\_index\_op);
                                                                                                                                                                                                                                                                                             ati_dff_in #(1) urb2_sx_index_op(sclk, RB2_SX_index_op,q_rb2_sx_index_op);
             input [0:0] RB0_SX_index_op,
RB3_SX_index_op;
                                                                                                RB1_SX_index_op,
                                                                                                                                                                      RB2_SX_index_op,
                                                                                                                                                                                                                                                                                             ati\_dff\_in~\#(1)~urb3\_sx\_index\_op(sclk, RB3\_SX\_index\_op,q\_rb3\_sx\_index\_op);\\
                wire~~[7:0] \qquad q\_rb0\_sx\_index,q\_rb1\_sx\_index,q\_rb2\_sx\_index,q\_rb3\_sx\_index;\\
12
                                                                                                                                                                                                                                                                                            ati_dff_out #(1) usx_rb0_index_rtr(sclk,sx_rb0_index_rtr, q_sx_rb0_index_rtr);
              q\_rb0\_sx\_index\_send, q\_rb1\_sx\_index\_send, q\_rb2\_sx\_index\_send, q\_rb3\_sx\_index\_send; \\
                                                                                                                                                                                                                                                                                             ati dff out #(1) usx rb1 index rtr(sclk,sx rb1 index rtr, q sx rb1 index rtr);
             wire [0:0] q_rb0_sx_index_op,
q_rb3_sx_index_op;
                                                                                                                                                                    q_rb2_sx_index_op,
                                                                                                q_rb1_sx_index_op,
                                                                                                                                                                                                                                                                                             ati_dff_out #(1) usx_rb2_index_rtr(sclk,sx_rb2_index_rtr, q_sx_rb2_index_rtr);
                                                                                                                                                                                                                                                                             15
                                                                                                                                                                                                                                                                                             ati dff out #(1) usx rb3 index rtr(sclk,sx rb3 index rtr, q sx rb3 index rtr);
17
                                                                                                                                                                                                                                                                             16
             output [0:0] SX_RB1_index_rtr,SX_RB2_index_rtr,SX_RB3_index_rtr;
                                                                                                                                                                        SX_RB0_index_rtr,
18
19
                                                                                                                                                                                                                                                                             17
                                                                                                                                                                                                                                                                                                                           SX RB0 index_rtr = q_sx_rb0_index_rtr;
                                                                                                                                                                                                                                                                                             assign
20
                                                                                                                                                                                                                                                                             18
                                                                                                                                                                                                                                                                                             assign
                                                                                                                                                                                                                                                                                                                           SX_RB1_index_rtr = q_sx_rb1_index_rtr;
21
                wire [0:0] sx rb0 index rtr, sx rb1 index rtr, sx rb2 index rtr, sx rb3 index rtr;
                                                                                                                                                                                                                                                                             19
                                                                                                                                                                                                                                                                                             assign
                                                                                                                                                                                                                                                                                                                           SX_RB2_index_rtr = q_sx_rb2_index_rtr;
                                           q_sx_rb0_index_rtr,
                                                                                                                                                                                                                                                                             20
                                                                                                                                                                                                                                                                                             assign
                                                                                                                                                                                                                                                                                                                           SX\_RB3\_index\_rtr = q\_sx\_rb3\_index\_rtr;
             q\_sx\_rb1\_index\_rtr, q\_sx\_rb2\_index\_rtr, q\_sx\_rb3\_index\_rtr;
23
                                                                                                                                                                                                                                                                             21
24
                                                                                                                                                                                                                                                                             22
25
                ati_dff_in #(1) urb0_sx_index_send(sclk, RB0_SX_index_send,q_rb0_sx_index_send);
                                                                                                                                                                                                                                                                             23
                                                                                                                                                                                                                                                                                             //PA(Primitive Assembly) - SX position export interface
26
                ati_dff_in #(1) urb1_sx_index_send(sclk, RB1_SX_index_send,q_rb1_sx_index_send);
                                                                                                                                                                                                                                                                             24
27
                ati\_dff\_in\ \#(1)\ urb2\_sx\_index\_send(sclk,\ RB2\_SX\_index\_send,q\_rb2\_sx\_index\_send);
                                                                                                                                                                                                                                                                             25
                                                                                                                                                                                                                                                                                                                           PA_SX_req;
                 ati\_dff\_in\ \#(1)\ urb3\_sx\_index\_send(sclk, RB3\_SX\_index\_send, q\_rb3\_sx\_index\_send);
                                                                                                   Page 17 of 25
                                                                                                                                                                                                                                                                                                                                                                               Page 18 of 25
                                                                                                                                                                                  Ex. 2109 - sx.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                                Ex. 2109 - sx.v
                                              PA SX sp id;
                input
                                                                                                                                                                                                                                                                               2
                input [1:0] PA SX offset; //used to be PA SX export phase
                                                                                                                                                                                                                                                                                                                           SX PA send = q sx pa send;
                                                                                                                                                                                                                                                                                             assign
                                             PA SX aux: //used to be PA SX 2ndbuff
                 input
                                                                                                                                                                                                                                                                                             assign
                                                                                                                                                                                                                                                                                                                           SX_PA_data = q_sx_pa_data;
                                             PA SX last:
                 input
                wire [0:0] q_pa_sx_req;
                 wire [0:0] q_pa_sx_sp_id;
                 wire [1:0] q_pa_sx_offset;
                                                                                                                                                                                                                                                                                                                  sx_id = unit_id;
                 wire [0:0] q_pa_sx_aux;
                 wire [0:0] q_pa_sx_last;
10
                                                                                                                                                                                                                                                                                             //.....Insert Logic.....
11
12
                ati_dff_in #(1) upa_sx_req(sclk,PA_SX_req,q_pa_sx_req);
                                                                                                                                                                                                                                                                                             parameter_caches uparam_caches(// Outputs
13
                 ati\_dff\_in\ \#(1)\ upa\_sx\_sp\_id(sclk,PA\_SX\_sp\_id,q\_pa\_sx\_sp\_id);
                                                                                                                                                                                                                                                                                          .SX_out_vtx_data0(SX_out_vtx_data0), .SX out vtx data1(SX out vtx data1), .SX out vtx data2(SX out vtx data2),
                 ati_dff_in #(2) upa_sx_offset(sclk,PA_SX_offset,q_pa_sx_offset);
                                                                                                                                                                                                                                                                                                                                                         .vtx data0(sx vtx data0),.vtx data1(sx vtx data1),
                 ati_dff_in #(1) upa_sx_aux(sclk,PA_SX_aux,q_pa_sx_aux);
                                                                                                                                                                                                                                                                                          .vtx data2(sx vtx data2),
16
                ati_dff_in #(1) upa_sx_last(sclk,PA_SX_last,q_pa_sx_last);
                                                                                                                                                                                                                                                                             18
17
                                                                                                                                                                                                                                                                                          . SQ\_SX\_ptr0(SQ\_SX\_pc\_ptr0), \\ . SQ\_SX\_ptr1(SQ\_SX\_pc\_ptr1), \\ . SQ\_SX\_ptr2(SQ\_SX\_pc\_ptr2), \\ . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), \\ . SQ\_SX
18
                                       SX_PA_send;
                output
19
                output [127:0] SX_PA_data;
                                                                                                                                                                                                                                                                                          .SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), SQ\_SX\_pc\_cmask(SQ\_SX\_pc\_channel\_mask), .SP\_SX\_data0(SP0\_SX\_data0), .SP\_SX\_data1(SP0\_SX\_data1), .SP\_SX\_data1(SP0\_SX\_data1), .SP\_SX\_data1), .SP\_SX\_data1(SP0\_SX\_data1), .SP\_SX\_data1), .SP\_SX\_data1), .SP\_SX\_data1, .SP\_SX\_data1), .SP\_SX_Data1), .SP\_SX_Data1
20
                                                                                                                                                                                                                                                                             24
21
                wire sx pa send, q sx pa send;
                                                                                                                                                                                                                                                                             25
                                                                                                                                                                                                                                                                             26
27
                                                                                                                                                                                                                                                                                          .SP_SX_data2(SP0_SX_data2),.SP_SX_data3(SP0_SX_data3),.SP_SX_data4(SP1_SX_data0), .SP_SX_data5(SP1_SX_data1),
22
                wire~[\,127:0]\quad sx\_pa\_data,~q\_sx\_pa\_data;
23
                                                                                                                                                                                                                                                                                          .sP\_SX\_data6(SP1\_SX\_data2),.SP\_SX\_data7(SP1\_SX\_data3),.sclk(sclk),.srst(srst),.sx\_id(sx\_id),.SX\_in\_vtx\_data0(SX\_in\_vtx\_data0),
24
                ati_dff_out #(1) usx_pa_send(sclk,sx_pa_send, q_sx_pa_send);
                 ati\_dff\_out~\#(128)~usx\_pa\_data(sclk,sx\_pa\_data,~q\_sx\_pa\_data);
                                                                                                                                                                                                                                                                                          .SX_in_vtx_data1(SX_in_vtx_data1),.SX_in_vtx_data2(SX_in_vtx_data2),
                                                                                                   Page 19 of 25
                                                                                                                                                                                                                                                                                                                                                                               Page 20 of 25
                                                                                                                                                                                   Ex. 2109 - sx.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                               Ex. 2109 - sx.v
```

```
.SQ SX interp flat vtx(SQ SX interp flat vtx),
                                                                                                                                                                                                                                                                                     .sx_pa_send(sx_pa_send), .sx_pa_data(sx_pa_data),
            SQ_SX_interp_flat_gouraud(SQ_SX_interp_flat_gouraud)
                                                                                                                                                                                                                                                                                     // Inputs
                                                                                                                                                                                                                                                                                      .sc_sx_quad_x(q_sc_sx_quad_x),
                                                                                                                                                                                                                                   .sc_sx_quad_y(q_sc_sx_quad_y),
                                                                                                                                                                                                                                  .sc\_sx\_quad\_mask(q\_sc\_sx\_quad\_mask), \\ .sc\_sx\_quad\_tilex(q\_sc\_sx\_quad\_tilex),
              export_control uexport_control(
                                                                                                                                                                                                                                  .sc\_sx\_quad\_tiley(q\_sc\_sx\_quad\_tiley), \\ .sc\_sx\_quad\_send(q\_sc\_sx\_quad\_send), \\
           .sx_sc_quad_rtr(sx_quad_rtr),
.sx_sq_exp_count_rdy(sx_sq_exp_count_rdy),
                                                                                                                                                                                                                                                                                     .sclk(sclk), .srst(srst),
                                                            //.SX_SQ_exp_pos_avail(sx_sq_exp_pos_avail),
10
                                                              .sx sq exp buf avail(sx sq exp buf avail),
                                                                                                                                                                                                                                  . sp0\_sx\_exp\_pvalid (q\_sp0\_sx\_exp\_pvalid), . sp1\_sx\_exp\_pvalid (q\_sp1\_sx\_exp\_pvalid), . sp0\_sx\_exp\_pvalid (q\_sp1\_sx\_exp\_pvalid), . sp0\_sx\_exp\_pvalid (q\_sp1\_sx\_exp\_pvalid), . sp0\_sx\_exp\_pvalid (q\_sp1\_sx\_exp\_pvalid), . sp1\_sx\_exp\_pvalid (q\_sp1\_sx\_exp\_pvalid (q\_sp1\_s
11
                                                              .sx rb quad x(sx rb quad x), .sx rb quad y(sx rb quad y),
                                                                                                                                                                                                                                  .sp0\_sx\_exp\_alu\_id(q\_sp0\_sx\_exp\_alu\_id), \\ .sp1\_sx\_exp\_alu\_id(q\_sp1\_sx\_exp\_alu\_id), \\
12
13
                                                               .sx_rb_quad_mask(sx_rb_quad_mask),
           .sx_rb_quad_type(sx_rb_quad_type).
                                                                                                                                                                                                                                  .sp0\_sx\_exporting(q\_sp0\_sx\_exporting), \\ .sp1\_sx\_exporting(q\_sp1\_sx\_exporting), \\
           .sx\_rb\_quad\_pixel(sx\_rb\_quad\_pixel),\\ .sx\_rb\_quad\_index(sx\_rb\_quad\_index),
14
15
                                                                                                                                                                                                                                  .sp0\_sx\_exp\_dest(q\_sp0\_sx\_exp\_dest), \\ .sp1\_sx\_exp\_dest(q\_sp1\_sx\_exp\_dest), \\
           .sx\_rb0\_quad\_send(sx\_rb0\_quad\_send), \\ .sx\_rb1\_quad\_send(sx\_rb1\_quad\_send), \\
                                                                                                                                                                                                                                 .sp0\_sx\_data0(q\_sp0\_sx\_data0), \\ .sp0\_sx\_data1(q\_sp0\_sx\_data1), \\
18
19
                                                                                                                                                                                                                                  .sp0\_sx\_data2(q\_sp0\_sx\_data2), \\ .sp0\_sx\_data3(q\_sp0\_sx\_data3), \\
           .sx_rb2_quad_send(sx_rb2_quad_send),.sx_rb3_quad_send(sx_rb3_quad_send),
                                                                                                                                                                                                                        20
21
           .sx_rb0_color_data(sx_rb0_color_data), .sx_rb1_color_data(sx_rb1_color_data),
20
21
                                                                                                                                                                                                                                  .sp1\_sx\_data0(q\_sp1\_sx\_data0), \\ .sp1\_sx\_data1(q\_sp1\_sx\_data1), \\
           .sx_rb2_color_data(sx_rb2_color_data), .sx_rb3_color_data(sx_rb3_color_data),
22
23
                                                                                                                                                                                                                                  .sp1\_sx\_data2(q\_sp1\_sx\_data2), \\ .sp1\_sx\_data3(q\_sp1\_sx\_data3),
           .sx_rb0_color_send(sx_rb0_color_send),.sx_rb1_color_send(sx_rb1_color_send),
                                                                                                                                                                                                                        26
27
                                                                                                                                                                                                                                  .sq sx exp type(q sq sx exp type),.sq sx exp number(q sq sx exp number),
           .sx\_rb2\_color\_send(sx\_rb2\_color\_send), \\ .sx\_rb3\_color\_send(sx\_rb3\_color\_send), \\
26
27
                                                                                                                                                                                                                                  .sq\_sx\_exp\_state(q\_sq\_sx\_exp\_state),.sq\_sx\_exp\_id(q\_sq\_sx\_exp\_id),\\
           .sx_rb0_index_rtr(sx_rb0_index_rtr), .sx_rb1_index_rtr(sx_rb1_index_rtr),
                                                                                                                                                                                                                        31
                                                                                                                                                                                                                                  .sq\_sx\_exp\_valid (q\_sq\_sx\_exp\_valid),.sq\_sx\_free\_done (q\_sq\_sx\_free\_done),\\
           .sx_rb2_index_rtr(sx_rb2_index_rtr),.sx_rb3_index_rtr(sx_rb3_index_rtr),
                                                                                                                                                                                                                                                                                     .sq\_sx\_free\_id(q\_sq\_sx\_free\_id),
                                                                              Page 21 of 25
                                                                                                                                                                                                                                                                                                      Page 22 of 25
                                                                                                                                               Ex. 2109 - sx.v
                                                                                                                                                                                                                                                                                                                                                                      Ex. 2109 - sx.v
           .rb0\_sx\_quad\_rtr(q\_rb0\_sx\_quad\_rtr),.rb1\_sx\_quad\_rtr(q\_rb1\_sx\_quad\_rtr),\\
           .rb2\_sx\_quad\_rtr(q\_rb2\_sx\_quad\_rtr),.rb3\_sx\_quad\_rtr(q\_rb3\_sx\_quad\_rtr),\\
                                                                                                                                                                                                                                    //registering the top level IO
           .rb0_sx_color_rtr(q_rb0_sx_color_rtr),.rb1_sx_color_rtr(q_rb1_sx_color_rtr),
           .rb2_sx_color_rtr(q_rb2_sx_color_rtr),.rb3_sx_color_rtr(q_rb3_sx_color_rtr),
                                                                                                                                                                                                                                     ati_dff_out #(128) usx_vtx_data0(sclk, sx_vtx_data0, q_sx_vtx_data0);
                                                                                                                                                                                                                                     ati\_dff\_out\ \#(132)\ usx\_vtx\_data1(sclk,\ sx\_vtx\_data1,\ q\_sx\_vtx\_data1);
                                                             .rb0_sx_index(q_rb0_sx_index),
                                                                                                                                                                                                                                     ati\_dff\_out\ \#(132)\ usx\_vtx\_data2(sclk,\ sx\_vtx\_data2,\ q\_sx\_vtx\_data2);
                                                              .rb1_sx_index(q_rb1_sx_index),
           .rb2_sx_index(q_rb2_sx_index),
11
           .rb3\_sx\_index(q\_rb3\_sx\_index), \\ .rb0\_sx\_index\_send(q\_rb0\_sx\_index\_send), \\
12
13
                                                                                                                                                                                                                        10
                                                                                                                                                                                                                                     assign
                                                                                                                                                                                                                                                           SX\_SP\_vtx\_data0 = q\_sx\_vtx\_data0;
                                                                                                                                                                                                                       11
                                                                                                                                                                                                                                                            SX_SP_vtx_datal = q_sx_vtx_datal;
           .rb1\_sx\_index\_send(q\_rb1\_sx\_index\_send), \\ .rb2\_sx\_index\_send(q\_rb2\_sx\_index\_send), \\
14
15
                                                                                                                                                                                                                       12
                                                                                                                                                                                                                                                            SX_SP_vtx_data2 = q_sx_vtx_data2;
16
                                                             .rb3\_sx\_index\_send(q\_rb3\_sx\_index\_send),
                                                                                                                                                                                                                       13
                                                                                                                                                                                                                        14
18
           .rb0_sx_index_op(q_rb0_sx_index_op),.rb1_sx_index_op(q_rb1_sx_index_op),
                                                                                                                                                                                                                       15
19
20
           .rb2 sx index op(q rb2 sx index op),.rb3 sx index op(q rb3 sx index op),
21
                                                              .rbbm sx soft reset(q rbbm sx soft reset),
22
                                                              .rbbm we(q rbbm we), .rbbm wd(q rbbm wd),
23
                                                              .rbbm_a(q_rbbm_a), .rbbm_be(q_rbbm_be),
24
                                                              .rbbm re(q rbbm re),
                                                                                                                                                                                                                        20
25
                                                              .rbb rs in(q rbb rs in), .rbb rd in(q rbb rd in),
                                                                                                                                                                                                                       21
26
                                                              .pa_sx_req(q_pa_sx_req), .pa_sx_sp_id(q_pa_sx_sp_id),
                                                                                                                                                                                                                        22
27
                                                              .pa_sx_offset(q_pa_sx_offset),.pa_sx_aux(q_pa_sx_aux),
                                                                                                                                                                                                                       23
28
                                                              .pa_sx_last(q_pa_sx_last)
                                                                                                                                                                                                                        24
29
                                                                                                                                                                                                                        25
                                                                              Page 23 of 25
                                                                                                                                                                                                                                                                                                      Page 24 of 25
                                                                                                                                               Ex. 2109 - sx.v
                                                                                                                                                                                                                                                                                                                                                                      Ex. 2109 - sx.v
```

```
1 //
                      -*- Mode: Verilog -*-
 2 // Filename : parameter caches.v
 3 // Description : This is a wrapper around 8 parameter cache memories that each SX has
                                                                                                                           input [10:0] SQ_SX_ptr0, SQ_SX_ptr1,SQ_SX_ptr2;
 4 // Author : Andi Skende
                                                                                                                     4 input SQ_SX_pc_wr_en;
 5 // Created On : Tue Mar 26 20:07:29 2002
                                                                                                                      5 input [6:0] SQ_SX_pc_wr_addr;
 6 // Last Modified By: .
                                                                                                                           input [3:0] SQ_SX_pc_cmask;
 7 // Last Modified On:
                                                                                                                          \label{eq:sps_scale} \begin{array}{ll} input \\ SP\ SX\ data0,SP\ SX\ data1,SP\ SX\ data2,SP\ SX\ data3,SP\ SX\ data4,SP\ SX\ data5,SP\ SX\ data7; \\ \end{array}
 8 // Update Count : 0
 9 // Status : Unknown, Use with caution!
                                                                                                                     10
                                                                                                                           input
                                                                                                                                      sclk, srst; //clock and reset
                                                                                                                     11
11 'timescale 1ns / 1ps
                                                                                                                     12
                                                                                                                           input [1:0] SQ_SX_interp_flat_vtx;
12 module parameter_caches
                                                                                                                    13
                                                                                                                           input [0:0] SQ_SX_interp_flat_gouraud;
13 (/*AUTOARG*/
14 // Outputs
                                                                                                                    15
                                                                                                                           output [127:0] SX_out_vtx_data0 , SX_out_vtx_data1 , SX_out_vtx_data2;
15
     SX_out_vtx_data0, SX_out_vtx_data1, SX_out_vtx_data2, vtx_data0,
                                                                                                                    16
                                                                                                                    17
                                                                                                                           output [127:0] vtx_data0;
17 // Inputs
                                                                                                                     18
                                                                                                                            output [131:0] vtx_data1, vtx_data2;
18 SQ SX ptr0, SQ SX ptr1, SQ SX ptr2, SQ SX pc wr en,
                                                                                                                    19
                                                                                                                           reg [127:0] vtx_data0_final, vtx_data1_final, vtx_data2_final;
19 SQ_SX_pc_wr_addr, SQ_SX_pc_cmask, SP_SX_data0, SP_SX_data1,
                                                                                                                    20
                                                                                                                           reg [127:0] q0_vtx_data0_final,q0_vtx_data1_final,q0_vtx_data2_final;
20 \qquad SP\_SX\_data2, SP\_SX\_data3, SP\_SX\_data4, SP\_SX\_data5, SP\_SX\_data6, \\
                                                                                                                    21
                                                                                                                           reg [127:0] q1_vtx_data0_final,q2_vtx_data0_final;
21
      SP SX data7, sclk, srst, sx id, SQ SX interp flat vtx,
                                                                                                                    22
                                                                                                                           reg [127:0] q1_vtx_data1_final,q2_vtx_data1_final;
      SQ\_SX\_interp\_flat\_gouraud, SX\_in\_vtx\_data0, SX\_in\_vtx\_data1,
23 SX in vtx data2
                                                                                                                     24
24
                                                                                                                     25
                                           Page 1 of 20
                                                                                                                                                                Page 2 of 20
                                                               Ex. 2110 - parameter caches.v
                                                                                                                                                                                    Ex. 2110 - parameter caches.v
                                                                                                                          wire [3:0] sx_pc0_id,sx_pc1_id,sx_pc2_id,sx_pc3_id,sx_pc4_id,sx_pc5_id,sx_pc6_id,sx_pc7_id;
      //----
      //input from the other sx-----
      input [127:0] SX_in_vtx_data0, SX_in_vtx_data1, SX_in_vtx_data2;
      reg [127:0] other vtx data0, other vtx data1, other vtx data2;
                                                                                                                           //creating an id for each of the parameter caches.
                                                                                                                            //this id will be used inside the module uparam_cache_ctl to decode the read pointers into the
      parameter [3:0] pc0_id = 4'h0,
                pc1 id = 4'h1,
                                                                                                                    11
                   pc2_id = 4'h2,
                                                                                                                     12
                                                                                                                                         sx pc0 id = sx id? pc4 id: pc0 id;
                                                                                                                           assign
11
                   pc3_id = 4'h3,
                                                                                                                     13
                                                                                                                            assign
                                                                                                                                         sx_pc1_id = sx_id ? pc5_id : pc1_id;
                   pc4_id = 4'h4,
                                                                                                                     14
                                                                                                                                          sx_pc2_id = sx_id ? pc6_id : pc2_id;
                  pc5_id = 4'h5,
                                                                                                                     15
                                                                                                                                         sx pc3 id = sx_id ? pc7_id : pc3_id;
                                                                                                                           assign
14
                pc6_id = 4'h6,
                                                                                                                           assign
                                                                                                                                         sx_pc4_id = sx_id ? pc12_id : pc8_id;
                   pc7_id = 4'h7;
15
                                                                                                                     17
                                                                                                                                         sx_pc5_id = sx_id ? pc13_id : pc9_id;
                                                                                                                     18
                                                                                                                           assign
                                                                                                                                         sx_pc6_id = sx_id ? pc14_id : pc10_id;
17
       parameter [3:0] pc8_id = 4'h8,
                                                                                                                     19
                                                                                                                                         sx_pe7_id = sx_id ? pc15_id : pc11_id;
                   pc9 id = 4'h9,
18
                   pc10_id = 4'ha,
                                                                                                                    21
20
                   pc11_id = 4'hb,
                   pc12 id = 4'hc,
21
                                                                                                                     23
                   pc13_id = 4'hd,
                                                                                                                     24 wire [127:0]
25 ovtx_data0,ovtx_data1,ovtx_data2,ovtx_data3,ovtx_data4,ovtx_data5,ovtx_data6,ovtx_data7;
23
                   pc14_id = 4'he,
                                                                                                                           reg [127:0] mux_vtx_data0, mux_vtx_data1, mux_vtx_data2;
                   pc15_id = 4'hf;
24
                                                                                                                           reg [127:0] q0_mux_vtx_data0, q0_mux_vtx_data1,q0_mux_vtx_data2;
                                           Page 3 of 20
                                                                                                                                                                Page 4 of 20
                                                               Ex. 2110 - parameter caches.v
                                                                                                                                                                                    Ex. 2110 - parameter caches.v
```

ATI 2110 LG v. ATI IPR2015-00326

```
reg [127:0] q1_mux_vtx_data0, q1_mux_vtx_data1,q1_mux_vtx_data2;
                                                                                                                                                                     .ovtx data(ovtx data1),
        reg\ [127:0] \qquad q2\_mux\_vtx\_data0,\ q2\_mux\_vtx\_data1,q2\_mux\_vtx\_data2;
                                                                                                                                                                     . SQ\_SX\_ptr0(SQ\_SX\_ptr2),. SQ\_SX\_ptr1(SQ\_SX\_ptr0), \\
                                                                                                                                     .SQ_SX_ptr2(SQ_SX_ptr1),
        reg\ [127:0] \qquad q3\_mux\_vtx\_data0,\ q3\_mux\_vtx\_data1, q3\_mux\_vtx\_data2;
        wire [131:0] vtx_diff10, vtx_diff20;
                                                                                                                                     . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en),. SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
        reg~[127:0] \qquad q0\_vtx\_diff10,~q0\_vtx\_diff20;
                                                                                                                                     .SQ_SX_pc_cmask(SQ_SX_pc_cmask),.SP_SX_data(SP_SX_data1),
       reg [127:0] vtx_provoking;
                                                                                                                                                                     .pc memory id(sx pcl id), .sclk(sclk),.srst(srst)
        reg [127:0] vtx_flat_gouraud0, vtx_flat_gouraud1,vtx_flat_gouraud2;
       reg \ [127:0] \qquad q0\_vtx\_flat\_gouraud0, \\ q0\_vtx\_flat\_gouraud1, \\ q0\_vtx\_flat\_gouraud2;
        reg [127:0] q1_vtx_flat_gouraud0, q2_vtx_flat_gouraud0;
                                                                                                                               11
                                                                                                                                      param cache ctl uparam cache ctl2(
       reg~[127:0] \qquad q1\_vtx\_flat\_gouraud2,~q2\_vtx\_flat\_gouraud2;
                                                                                                                               12
                                                                                                                                                                     .ovtx data(ovtx data2),
11
       reg \ [127:0] \qquad q1\_vtx\_flat\_gouraud1, \ q2\_vtx\_flat\_gouraud1;
                                                                                                                                                                     . SQ\_SX\_ptr0(SQ\_SX\_ptr2),. SQ\_SX\_ptr1(SQ\_SX\_ptr0), \\
12
                                                                                                                                     .SQ_SX_ptr2(SQ_SX_ptr1),
13
                                                                                                                                     . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), . SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
14
       param cache ctl uparam cache ctl0(
15
                                     .ovtx data(ovtx data0),
                                                                                                                                     .SQ_SX_pc_cmask(SQ_SX_pc_cmask),.SP_SX_data(SP_SX_data2),
                                      . SQ\_SX\_ptr0(SQ\_SX\_ptr2),. SQ\_SX\_ptr1(SQ\_SX\_ptr0), \\
16
17
                                                                                                                               19
                                                                                                                                                                     .pc_memory_id(sx_pc2_id), .sclk(sclk),.srst(srst)
      .SQ SX ptr2(SQ SX ptr1),
                                                                                                                               20
                                                                                                                                                                    );
      . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), . SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
                                                                                                                               21
                                                                                                                                       param cache ctl uparam cache ctl3(
20
21
                                                                                                                               22
                                                                                                                                                                     .ovtx data(ovtx data3),
      .SQ_SX_pc_cmask(SQ_SX_pc_cmask),.SP_SX_data(SP_SX_data0),
                                                                                                                                                                     .SQ_SX_ptr0(SQ_SX_ptr2),.SQ_SX_ptr1(SQ_SX_ptr0),
                                                                                                                               23
24
22
                                      .pc_memory_id(sx_pc0_id), .sclk(sclk),.srst(srst)
                                                                                                                                     .SQ SX ptr2(SQ SX ptr1),
23
                                                                                                                                     . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), . SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
24
25
                                                                                                                                     . SQ\_SX\_pc\_cmask (SQ\_SX\_pc\_cmask), . SP\_SX\_data (SP\_SX\_data3), \\
       param cache ctl uparam cache ctl1(
                                                                                                                                                                     .pc_memory_id(sx_pc3_id), .sclk(sclk),.srst(srst)
                                               Page 5 of 20
                                                                                                                                                                              Page 6 of 20
                                                                     Ex. 2110 - parameter_caches.v
                                                                                                                                                                                                   Ex. 2110 - parameter_caches.v
                                                                                                                                                                     .pc memory id(sx pc6 id), .sclk(sclk),.srst(srst)
 2
       param cache ctl uparam cache ctl4(
                                      .ovtx data(ovtx data4).
                                                                                                                                       param cache ctl uparam cache ctl7(
                                      . SQ\_SX\_ptr0(SQ\_SX\_ptr2),. SQ\_SX\_ptr1(SQ\_SX\_ptr0), \\
                                                                                                                                                                     .ovtx data(ovtx data7),
      .SQ SX ptr2(SQ SX ptr1),
                                                                                                                                                                     . SQ\_SX\_ptr0(SQ\_SX\_ptr2),. SQ\_SX\_ptr1(SQ\_SX\_ptr0), \\
                                                                                                                                     .SQ_SX_ptr2(SQ_SX_ptr1),
      . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), . SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
                                                                                                                                     . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), . SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
      .SQ_SX_pc_cmask(SQ_SX_pc_cmask),.SP_SX_data(SP_SX_data4),
                                     .pc_memory_id(sx_pc4_id), .sclk(sclk),.srst(srst)
                                                                                                                                     . SQ\_SX\_pc\_cmask(SQ\_SX\_pc\_cmask),. SP\_SX\_data(SP\_SX\_data7), \\
11
                                                                                                                                                                     .pc_memory_id(sx_pc7_id), .sclk(sclk),.srst(srst)
       param_cache_ctl uparam_cache_ctl5(
13
                                     .ovtx data(ovtx data5),
                                     . SQ\_SX\_ptr0(SQ\_SX\_ptr2),. SQ\_SX\_ptr1(SQ\_SX\_ptr0), \\
14
15
      .SQ SX ptr2(SQ SX ptr1),
                                                                                                                               15
                                                                                                                                    //Selecting three vertex vectors-----
      . SQ\_SX\_pc\_wr\_en(SQ\_SX\_pc\_wr\_en), . SQ\_SX\_pc\_wr\_addr(SQ\_SX\_pc\_wr\_addr), \\
                                                                                                                               16
                                                                                                                               17
      .SQ_SX_pc_cmask(SQ_SX_pc_cmask),.SP_SX_data(SP_SX_data5),
                                                                                                                               18
                                                                                                                                      reg [3:0]
                                                                                                                                                   vtx sel0, vtx sel1, vtx sel2;
20
                                      .pc_memory_id(sx_pc5_id), .sclk(sclk),.srst(srst)
                                                                                                                               19
                                                                                                                                       reg [3:0]
                                                                                                                                                     q0_vtx_sel0, q0_vtx_sel1, q0_vtx_sel2;
21
                                                                                                                                       reg [3:0]
                                                                                                                                                     q1_vtx_sel0, q1_vtx_sel1, q1_vtx_sel2;
       param_cache_ctl uparam_cache_ctl6(
                                                                                                                               21
                                                                                                                                       reg [3:0]
                                                                                                                                                     q2\_vtx\_sel0,\,q2\_vtx\_sel1,\,q2\_vtx\_sel2;
23
                                      .ovtx_data(ovtx_data6),
                                                                                                                               22
                                                                                                                                       reg [3:0]
                                                                                                                                                     q3\_vtx\_sel0,\,q3\_vtx\_sel1,\,q3\_vtx\_sel2;\\
                                     .SQ_SX_ptr0(SQ_SX_ptr2),.SQ_SX_ptr1(SQ_SX_ptr0),
24
25
                                                                                                                              23
      .SQ_SX_ptr2(SQ_SX_ptr1),
                                                                                                                               24
                                                                                                                                      reg [127:0] vertex0, vertex1,vertex2;
26
27
      .SQ_SX_pc_wr_en(SQ_SX_pc_wr_en),.SQ_SX_pc_wr_addr(SQ_SX_pc_wr_addr),
                                                                                                                               25
                                                                                                                               26
      .SQ_SX_pc_cmask(SQ_SX_pc_cmask),.SP_SX_data(SP_SX_data6),
                                               Page 7 of 20
                                                                                                                                                                              Page 8 of 20
                                                                     Ex. 2110 - parameter caches.v
                                                                                                                                                                                                   Ex. 2110 - parameter caches.v
```

```
reg [1:0]
                                    flat_vtx_sel, q0_flat_vtx_sel, q1_flat_vtx_sel,q2_flat_vtx_sel,q3_flat_vtx_sel;
                                                                                                                                                                                                                                         flat_vtx_sel<= SQ_SX_interp_flat_vtx;
          reg \ [0.0] \\ flat \ gouraud \ sel, q0 \ flat \ gouraud \ sel, q1 \ flat \ gouraud \ sel, q2 \ flat \ gouraud \ sel, q3 \ flat \ gouraud \ sel, q2 \ flat \ gouraud \ sel, q3 \ flat \ gouraud \ sel, q4 \ flat \ gouraud \ sel, q5 \ flat \ gouraud \ sel, q6 \ flat \ gouraud \ sel, q8 \ flat \ gouraud \ sel, q9 \ flat 
                                                                                                                                                                                                                                         q0\_flat\_vtx\_sel \! <= 2"b0;
                                                                                                                                                                                                                                         q1\_flat\_vtx\_sel <= 2"b0;
                                                                                                                                                                                                                                         q2\_flat\_vtx\_sel <= 2"b0;
                                                                                                                                                                                                                                         q3\_flat\_vtx\_sel \mathrel{<=} 2"b0;
             always @(posedge sclk)
                                                                                                                                                                                                                                         flat_gouraud_sel <= 1'b0;
                                                                                                                                                                                                                                         q0\_flat\_gouraud\_sel <= 1"b0";
              begin
                                                                                                                                                                                                                                         q1_flat_gouraud_sel <= 1'b0;
                     if(srst)
                                                                                                                                                                                                                                         q2_flat_gouraud_sel <= 1'b0;
10
                       begin
11
                                                                                                                                                                                                                                         q3\_flat\_gouraud\_sel <= 1"b0";
                                                                                                                                                                                                               11
12
                          vtx sel0<= 4'b0;
                                                                                                                                                                                                               12
13
                          vtx sel1<= 4'b0;
                                                                                                                                                                                                               13
                                                                                                                                                                                                                                      begin
14
                          vtx sel2<= 4'b0:
                                                                                                                                                                                                               14
                                                                                                                                                                                                                                         vtx_sel0 <= SQ_SX_ptr0[10:7];
15
                          q0 vtx sel0<= 4'b0;
                                                                                                                                                                                                               15
                                                                                                                                                                                                                                         vtx sel1 <= SQ SX ptr1[10:7];
16
                          q0_vtx_sel1<= 4'b0;
                                                                                                                                                                                                               16
                                                                                                                                                                                                                                         vtx sel2 <= SQ SX ptr2[10:7];
17
                          q0_vtx_sel2<= 4'b0;
                                                                                                                                                                                                               17
18
                          q1\_vtx\_sel0 \! <= 4 b0;
                                                                                                                                                                                                                                         other vtx data0 <= SX in vtx data0:
                                                                                                                                                                                                               18
                                                                                                                                                                                                                                         other vtx data1 <= SX in vtx data1:
19
                          q1\_vtx\_sel1 \! <= 4"b0;
20
                          q1_vtx_sel2<= 4'b0;
                                                                                                                                                                                                               19
                                                                                                                                                                                                                                         other vtx data2 <= SX in vtx data2;
21
                          q2_vtx_sel0<= 4'b0;
                                                                                                                                                                                                               20
                                                                                                                                                                                                                                         q0 vtx sel0 <= vtx sel0;
                          q2\_vtx\_sel1 <= 4'b0;
22
                                                                                                                                                                                                               21
                                                                                                                                                                                                                                         q0\_vtx\_sel1 \mathrel{<=} vtx\_sel1;
                          q2_vtx_sel2<= 4'b0;
23
                                                                                                                                                                                                               22
                                                                                                                                                                                                                                         q0\_vtx\_sel2 \mathrel{<=} vtx\_sel2;
24
                          q3_vtx_sel0<= 4'b0;
                                                                                                                                                                                                               23
                                                                                                                                                                                                                                         q1\_vtx\_sel0 \mathrel{<=} q0\_vtx\_sel0;
                          q3_vtx_sel1<= 4'b0;
                                                                                                                                                                                                               24
                                                                                                                                                                                                                                         q1\_vtx\_sel1 \mathrel{<=} q0\_vtx\_sel1;
                                                                                                                                                                                                                                         q1\_vtx\_sel2 \mathrel{<=} q0\_vtx\_sel2;
                          q3_vtx_sel2<= 4'b0;
                                                                            Page 9 of 20
                                                                                                                                                                                                                                                                                          Page 10 of 20
                                                                                                                Ex. 2110 - parameter_caches.v
                                                                                                                                                                                                                                                                                                                               Ex. 2110 - parameter_caches.v
                          q2_vtx_sel0 <= q1_vtx_sel0 ;
                                                                                                                                                                                                                                         flat_vtx_sel<= 2'b0;
                          q2 vtx sell <= q1 vtx sell;
                                                                                                                                                                                                                                         q0 flat vtx sel<=flat vtx sel;
                          q2\_vtx\_sel2 \le q1\_vtx\_sel2;
                                                                                                                                                                                                                                         q1\_flat\_vtx\_sel \mathrel{<=} q0\_flat\_vtx\_sel;
                          q3 \text{ vtx sel0} \le q2 \text{ vtx sel0};
                                                                                                                                                                                                                                         q2\_flat\_vtx\_sel \mathrel{<=} q1\_flat\_vtx\_sel;
                          q3_vtx_sel1 \le q2_vtx_sel1;
                                                                                                                                                                                                                                         q3_flat_vtx_sel \le q2_flat_vtx_sel;
                          q3_vtx_sel2 \le q2_vtx_sel2;
                                                                                                                                                                                                                                         flat_gouraud_sel <= SQ_SX_interp_flat_gouraud;
                          q0\_mux\_vtx\_data0 \mathrel{<=} mux\_vtx\_data0;
                                                                                                                                                                                                                                         q0\_flat\_gouraud\_sel \mathrel{<=} flat\_gouraud\_sel;
                          q0\_mux\_vtx\_data1 \mathrel{<=} mux\_vtx\_data1;
                                                                                                                                                                                                                                         q\,l\_flat\_gouraud\_sel <= q0\_flat\_gouraud\_sel;
                          q0\_mux\_vtx\_data2 \mathrel{<=} mux\_vtx\_data2;
                                                                                                                                                                                                                                         q2\_flat\_gouraud\_sel <= q1\_flat\_gouraud\_sel \ ;
10
                          q1\_mux\_vtx\_data0 \mathrel{<=} q0\_mux\_vtx\_data0;
                                                                                                                                                                                                               10
                                                                                                                                                                                                                                         q3\_flat\_gouraud\_sel \mathrel{<=} q2\_flat\_gouraud\_sel;
11
                          q1\_mux\_vtx\_data1 \mathrel{<=} q0\_mux\_vtx\_data1;
                                                                                                                                                                                                               11
                                                                                                                                                                                                                                         q0\_vtx\_flat\_gouraud0 \mathrel{<=} vtx\_flat\_gouraud0;
12
                          q1\_mux\_vtx\_data2 <= q0\_mux\_vtx\_data2;
                                                                                                                                                                                                               12
                                                                                                                                                                                                                                         q0\_vtx\_flat\_gouraud1 \mathrel{<=} vtx\_flat\_gouraud1;
13
                                                                                                                                                                                                               13
                          q2\_mux\_vtx\_data0 <= q1\_mux\_vtx\_data0;
                                                                                                                                                                                                                                         q1\_vtx\_flat\_gouraud1 \mathrel{<=} q0\_vtx\_flat\_gouraud1;
14
                                                                                                                                                                                                               14
                          q2\_mux\_vtx\_data1 \mathrel{<=} q1\_mux\_vtx\_data1;
                                                                                                                                                                                                                                         q2\_vtx\_flat\_gouraud1 \mathrel{<=} q1\_vtx\_flat\_gouraud1;
15
                          q2\_mux\_vtx\_data2 <= q1\_mux\_vtx\_data2;
                                                                                                                                                                                                                                         q0\_vtx\_flat\_gouraud2 <= vtx\_flat\_gouraud2;
                          q3\_mux\_vtx\_data0 <= q2\_mux\_vtx\_data0;
                                                                                                                                                                                                                                         q1\_vtx\_flat\_gouraud2 \mathrel{<=} q0\_vtx\_flat\_gouraud2;
17
                          q3\_mux\_vtx\_data1 <= q2\_mux\_vtx\_data1;
                                                                                                                                                                                                                                         q2\_vtx\_flat\_gouraud2 <= q1\_vtx\_flat\_gouraud2;
18
                                                                                                                                                                                                               18
                                                                                                                                                                                                                                         q1 vtx flat gouraud0 <= q0 vtx flat gouraud0;
                          q3 mux vtx data2 <= q2 mux vtx data2;
19
                          q0 vtx data0 final <= vtx data0 final;
                                                                                                                                                                                                                                         q2 vtx flat gouraud0 <= q1 vtx flat gouraud0;
20
                          q1 vtx data0 final <= q0 vtx data0 final;
                                                                                                                                                                                                               20
                                                                                                                                                                                                                                      end // else: !if(srst)
21
                          q2 vtx data0 final <= q1 vtx data0 final;
                                                                                                                                                                                                               21
                                                                                                                                                                                                                             end // always @ (posedge sclk)
22
                                                                                                                                                                                                               22
                          q0 vtx data1 final <= vtx data1 final;
23
                                                                                                                                                                                                               23
                          q0 vtx data2 final <= vtx data2 final;
                          q0 vtx diff10 <= vtx diff10:
24
                                                                                                                                                                                                               24
                                                                                                                                                                                                                           //three 8:1 muxes used to select the vertices
                          q0 vtx diff20 <= vtx diff20;
                                                                                                                                                                                                                            //the 8:1 muxes are followed by a 2:1 mux each to select from this SX data
                                                                            Page 11 of 20
                                                                                                                                                                                                                                                                                          Page 12 of 20
                                                                                                                Ex. 2110 - parameter caches.v
                                                                                                                                                                                                                                                                                                                               Ex. 2110 - parameter caches.v
```

```
//or the "other" data coming from the other SX
                                                                                                                                        or sx_pc2_id or sx_pc3_id or sx_pc4_id or sx_pc5_id
2
                                                                                                                           2
                                                                                                                                        or sx_pc6_id or sx_pc7_id)
 3
       always@(/*AUTOSENSE*/ovtx_data0 or ovtx_data1 or ovtx_data2
                                                                                                                           3
                                                                                                                                  begin
 4
              or ovtx_data3 or ovtx_data4 or ovtx_data5 or ovtx_data6
                                                                                                                           4
                                                                                                                                      case(q1_vtx_sel1)
              or ovtx_data7 or q1_vtx_sel0 or sx_pc0_id or sx_pc1_id
                                                                                                                                       sx_pc0_id:mux_vtx_data1 = ovtx_data0;
              or sx_pc2_id or sx_pc3_id or sx_pc4_id or sx_pc5_id
                                                                                                                                       sx_pcl_id:mux_vtx_data1 = ovtx_data1;
                                                                                                                                       sx_pc2_id:mux_vtx_data1 = ovtx_data2;
              or sx_pc6_id or sx_pc7_id)
                                                                                                                                       sx_pc3_id:mux_vtx_data1 = ovtx_data3;
            case(q1_vtx_sel0)
                                                                                                                                       sx_pc4_id:mux_vtx_data1 = ovtx_data4;
10
             sx\_pc0\_id:mux\_vtx\_data0 = ovtx\_data0;
                                                                                                                                       sx_pc5_id:mux_vtx_data1 = ovtx_data5;
11
             sx_pcl_id:mux_vtx_data0 = ovtx_data1;
                                                                                                                                       sx_pc6_id:mux_vtx_data1 = ovtx_data6;
12
             sx_pc2_id:mux_vtx_data0 = ovtx_data2;
                                                                                                                                       sx_pc7_id:mux_vtx_data1 = ovtx_data7;
13
             sx_pc3_id:mux_vtx_data0 = ovtx_data3;
                                                                                                                                       default : mux_vtx_data1 = ovtx_data0;
14
             sx_pc4_id:mux_vtx_data0 = ovtx_data4;
                                                                                                                                      endcase // case(q1_vtx_sel1[2:0])
15
             sx pc5 id:mux vtx data0 = ovtx data5;
                                                                                                                         15
                                                                                                                                  end // always@ (q1 vtx sell or ovtx data0 or ovtx data1 or ovtx data2 or...
             sx pc6 id:mux vtx data0 = ovtx data6;
                                                                                                                          16
16
17
                                                                                                                         17
             sx pc7 id:mux vtx data0 = ovtx data7:
                                                                                                                                 always@(/*AUTOSENSE*/ovtx data0 or ovtx data1 or ovtx data2
18
             default : mux vtx data0 = ovtx data0:
                                                                                                                         18
                                                                                                                                        or ovtx data3 or ovtx data4 or ovtx data5 or ovtx data6
19
            endcase // case(q0 vtx sel0[2:0])
                                                                                                                          19
                                                                                                                                        or ovtx data7 or q1 vtx sel2 or sx pc0 id or sx pc1 id
20
         end // always@ (q0 vtx sel0 or ovtx data0 or ovtx data1 or ovtx data2 or.
                                                                                                                         20
                                                                                                                                        or sx pc2 id or sx pc3 id or sx pc4 id or sx pc5 id
21
                                                                                                                         21
                                                                                                                                        or sx_pc6_id or sx_pc7_id)
22
                                                                                                                         22
                                                                                                                                  begin
23
        always@(/*AUTOSENSE*/ovtx_data0 or ovtx_data1 or ovtx_data2
                                                                                                                         23
                                                                                                                                      case(q1 vtx sel2)
24
              or ovtx_data3 or ovtx_data4 or ovtx_data5 or ovtx_data6
                                                                                                                         24
                                                                                                                                        sx_pc0_id:mux_vtx_data2 = ovtx_data0;
25
              or ovtx_data7 or q1_vtx_sel1 or sx_pc0_id or sx_pc1_id
                                                                                                                         25
                                                                                                                                        sx_pc1_id:mux_vtx_data2 = ovtx_data1;
                                            Page 13 of 20
                                                                                                                                                                      Page 14 of 20
                                                                  Ex. 2110 - parameter_caches.v
                                                                                                                                                                                            Ex. 2110 - parameter_caches.v
             sx_pc2_id:mux_vtx_data2 = ovtx_data2;
                                                                                                                                  always @(/*AUTOSENSE*/final_sel0 or final_sel1 or other_vtx_data1
                                                                                                                           2
                                                                                                                                         or q0_mux_vtx_data1 or q2_vtx_sel1)
             sx pc3 id:mux vtx data2 = ovtx data3;
             sx pc4 id:mux vtx data2 = ovtx data4;
                                                                                                                                  begin
             sx pc5 id:mux vtx data2 = ovtx data5;
                                                                                                                                      case(q2 vtx sel1[2])
             sx_pc6_id:mux_vtx_data2 = ovtx_data6;
                                                                                                                                        final_sel0:vtx_data1_final = q0_mux_vtx_data1;
             sx_pc7_id:mux_vtx_data2 = ovtx_data7;
                                                                                                                                        final_sel1:vtx_data1_final = other_vtx_data1;
             default: mux\_vtx\_data2 = ovtx\_data0;
                                                                                                                           7
                                                                                                                                      endcase \textit{//} case(q1\_vtx\_sel1[2])
            endcase // case(q1 vtx sel2[2:0])
                                                                                                                           8
         end // always@ (q1_vtx_sel2 or ovtx_data0 or ovtx_data1 or ovtx_data2 or...
                                                                                                                                  always @(/*AUTOSENSE*/final_sel0 or final_sel1 or other_vtx_data2
10
                                                                                                                         10
11
       wire [0:0] final_sel0;
                                                                                                                         11
                                                                                                                                         or q0_mux_vtx_data2 or q2_vtx_sel2)
12
        wire [0:0] final_sel1;
                                                                                                                         12
13
                                                                                                                         13
                                                                                                                                      case(q2_vtx_sel2[2])
14
       assign final_sel0 = sx_id ? 1'b1: 1'b0;
                                                                                                                         14
                                                                                                                                       final\_sel0:vtx\_data2\_final = q0\_mux\_vtx\_data2;
15
                                                                                                                         15
                                                                                                                                        final_sel1:vtx_data2_final = other_vtx_data2;
       assign final_sel1 = ~ final_sel0;
16
                                                                                                                                      end case \textit{//} case(q1\_vtx\_sel2[2])
17
       always @(/*AUTOSENSE*/final_sel0 or final_sel1 or other_vtx_data0
                                                                                                                         17
                                                                                                                                  end
18
                                                                                                                          18
              or q0 mux vtx data0 or q2 vtx sel0)
19
        begin
20
            case(q2 vtx sel0[2])
                                                                                                                         20
21
             final sel0:vtx data0 final = q0 mux vtx data0;
                                                                                                                         21
                                                                                                                                 //Flat vs. Gouraud shading vertex selection...two level muxing
22
                                                                                                                         22
             final sell:vtx data0 final = other vtx data0:
23
                                                                                                                         23
            endcase // case(q1 vtx sel0[2])
24
        end
                                                                                                                         24
                                                                                                                                 always @(/*AUTOSENSE*/q0 vtx data0 final or q0 vtx data1 final
25
                                                                                                                         2.5
                                                                                                                                        or q0 vtx data2 final or q3 flat vtx sel)
                                            Page 15 of 20
                                                                                                                                                                      Page 16 of 20
                                                                  Ex. 2110 - parameter caches.v
                                                                                                                                                                                            Ex. 2110 - parameter caches.v
```

```
1
         begin
                                                                                                                                                end
 2
              case(q3\_flat\_vtx\_sel)
                                                                                                                                       2
              2'b00:vtx_provoking = q0_vtx_data0_final;
                                                                                                                                       3
                                                                                                                                              always @(/*AUTOSENSE*/q0_vtx_data2_final or q3_flat_gouraud_sel
               2'b01:vtx_provoking = q0_vtx_data1_final;
                                                                                                                                       4
               2'b10:vtx\_provoking = q0\_vtx\_data2\_final;
               default: vtx_provoking = q0_vtx_data0_final;
                                                                                                                                                    case(q3\_flat\_gouraud\_sel)
              endcase \textit{//} case(SQ\_SP\_interp\_flat\_vtx)
                                                                                                                                                     1'b0:vtx\_flat\_gouraud2 = vtx\_provoking \ ;
                                                                                                                                                     l'b1:vtx\_flat\_gouraud2 = q0\_vtx\_data2\_final \; ;
                                                                                                                                                   endcase // case(SQ_SP_interp_flat_gouraud)
10
         always @(/*AUTOSENSE*/q0_vtx_data0_final or q3_flat_gouraud_sel
                                                                                                                                      10
11
                                                                                                                                      11
                or vtx_provoking)
12
         begin
                                                                                                                                      12
13
                                                                                                                                      13
              case(q3 flat gouraud sel)
14
               1'b0:vtx_flat_gouraud0 =vtx_provoking ;
                                                                                                                                      14
15
                                                                                                                                              //Difference engine for the vertex attributes. This difference is used by the barycentric
15
               1'b1:vtx flat gouraud0 =q0 vtx data0 final;
                                                                                                                                      16
16
              endcase // case(SQ SP interp flat gouraud)
                                                                                                                                      17
                                                                                                                                              wire [32:0] vertex10_red, vertex10_green, vertex10_blue, vertex10_alpha;
17
                                                                                                                                      18
                                                                                                                                              wire [32:0] vertex20_red, vertex20_green, vertex20_blue, vertex20_alpha;
18
                                                                                                                                      19
                                                                                                                                           param_sub param10_red(q0_vtx_flat_gouraud1[31:0], q0_vtx_flat_gouraud0[31:0], vertex10_red, sclk);
19
        always @(/*AUTOSENSE*/q0_vtx_data1_final or q3_flat_gouraud_sel
                                                                                                                                      20
21
20
                or vtx provoking)
                                                                                                                                           param\_sub\_param10\_green(q0\_vtx\_flat\_gouraud1[63:32], \quad q0\_vtx\_flat\_gouraud0[63:32], \\ vertex10\_green, selk);
21
         begin
22
              case(q3\_flat\_gouraud\_sel)
                                                                                                                                             param\_sub \quad param10\_blue(q0\_vtx\_flat\_gouraud1[95:64], \quad q0\_vtx\_flat\_gouraud0[95:64], \\ vertex10\_blue, sclk);
23
               1'b0:vtx_flat_gouraud1 =vtx_provoking ;
24
              l'b1:vtx_flat_gouraud1 =q0_vtx_data1_final;
                                                                                                                                             param\_sub\_param10\_alpha(q0\_vtx\_flat\_gouraud1[127:96], \quad q0\_vtx\_flat\_gouraud0[127:96], \\ vertex10\_alpha, sclk);
25
              endcase // case(SQ_SP_interp_flat_gouraud)
                                                 Page 17 of 20
                                                                                                                                                                                       Page 18 of 20
                                                                         Ex. 2110 - parameter_caches.v
                                                                                                                                                                                                               Ex. 2110 - parameter_caches.v
                                                                                                                                              assign vtx data1 = q0 vtx diff10;
      \begin{array}{lll} param\_sub & param20\_red(q0\_vtx\_flat\_gouraud2[31:0], & q0\_vtx\_flat\_gouraud0[31:0], \\ vertex20\_red, sclk); \end{array}
                                                                                                                                              assign vtx data2 = q0 vtx diff20;
       param\_sub\_param20\_green(q0\_vtx\_flat\_gouraud2[63:32], \quad q0\_vtx\_flat\_gouraud0[63:32], \\ vertex20\_green, sclk);
       param_sub param20_blue(q0_vtx_flat_gouraud2[95:64], q0_vtx_flat_gouraud0[95:64], vertex20_blue, sclk);
                                                                                                                                       6 endmodule // parameter caches
       param\_sub \quad param20\_alpha(q0\_vtx\_flat\_gouraud2[127:96], \quad q0\_vtx\_flat\_gouraud0[127:96], \quad vertex20\_alpha, sclk);
11
12
13
                                                                                                                                      12
14
        //Outputs to the other SX block
15
16
        assign SX out vtx data0 = mux vtx data0;
17
         assign SX out vtx data1 = mux vtx data1;
18
         assign\ SX\_out\_vtx\_data2 = mux\_vtx\_data2;
19
20
21
22
23
24
25
        assign\ vtx\_diff10 = \{vertex10\_alpha, vertex10\_blue, vertex10\_green, vertex10\_red\};
26
         assign vtx_diff20 = {vertex20_alpha,vertex20_blue,vertex20_green,vertex20_red};
         assign vtx_data0 = q2_vtx_flat_gouraud0;
                                                 Page 19 of 20
                                                                                                                                                                                       Page 20 of 20
                                                                         Ex. 2110 - parameter caches.v
                                                                                                                                                                                                               Ex. 2110 - parameter caches.v
```

```
1 //
                    -*- Mode: Verilog -*-
                                                                                                             2 input [10:0] SQ SX ptr0, SQ SX ptr1,SQ SX ptr2;
 2 // Filename : param cache ctl.v
 3 // Description : This module implements the read/write logic for the parameter caches
                                                                                                            3 input SQ_SX_pc_wr_en;
 4 // Author : Andi Skende
                                                                                                            4 input [6:0] SQ_SX_pc_wr_addr;
 5 // Created On : Tue Mar 26 19:57:49 2002
                                                                                                            5 input [3:0] SO SX pc cmask:
                                                                                                                 input [127:0] SP_SX_data;
 7 // Last Modified On:
                                                                                                                  input [3:0] pc_memory_id; //0...16 a different id for each instance.
 8 // Update Count : 0
                                                                                                                               //The first 8 PCs 0-7 belong to SX0, while 8-15 to SX1
 9 // Status : Unknown, Use with caution!
                                                                                                                  input sclk, srst; //clock and reset
                                                                                                            10
11 'timescale 1ns / 1ps
                                                                                                            11
                                                                                                            12
13 module param_cache_ctl
                                                                                                            13
                                                                                                                 //Data out of the parameter caches
                                                                                                            14
14 (/*AUTOARG*/
15
                                                                                                            15
     // Outputs
                                                                                                            16
                                                                                                                  output [127:0] ovtx_data;
17 // Inputs
                                                                                                            17
18 SQ_SX_ptr0, SQ_SX_ptr1, SQ_SX_ptr2, SQ_SX_pc_wr_en,
                                                                                                            18 reg [10:0] pc_ptr0, pc_ptr1,pc_ptr2;
                                                                                                            19 reg
19 SQ_SX_pc_wr_addr, SQ_SX_pc_cmask, SP_SX_data, pc_memory_id, sclk,
                                                                                                                          pc_wr_en;
20 srst
                                                                                                            20 reg [6:0] pc_wr_addr;
                                                                                                            21 reg [3:0] pc_cmask;
21 );
                                                                                                                 reg [127:0] vertex_data_in;
23
                                                                                                            23 wire [127:0] vertex data out;
                                                                                                            24 reg [127:0] q0_vertex_data_out;
24
     //PC control and interpolation interface
                                       Page 1 of 14
                                                                                                                                                    Page 2 of 14
                                                                                                                                                                        Ex. 2111 - param cache ctl.v
                                                           Ex. 2111 - param cache ctl.v
 1 wire [3:0] mem_id;
                                                                                                                          q0_vertex_data_out <= vertex_data_out;
 2 assign mem_id = pc_memory_id;
                                                                                                                        end
                                                                                                                   end // always @ (posedge sclk)
 4
      always @(posedge sclk)
       begin
                                                                                                             7 //parameter cache write address decoding
           pc_ptr0 <= 11'b0;
                                                                                                            8 //finding out whether any of the above addresses applies to this parameter cache memory
           pc_ptr1 <= 11'b0;
                                                                                                                  //in other words if the vertex0 1 or 2 data is in this specific memory
            pc_ptr2 <= 11'b0;
11
            pc_wr_en <= 1'b0;
                                                                                                            11
           pc_wr_addr <= 7*b0;
                                                                                                            12 wire is_ptr0, is_ptr1,is_ptr2;
12
           pc_cmask <= 4'b0;
                                                                                                            13
14
            vertex data in <= 127b0;
                                                                                                            14
            q0_vertex_data_out <= 127'b0;
                                                                                                            15
15
                                                                                                                 assign is ptr0 = &(pc ptr0[10:7] \cdot mem id);
                                                                                                                  assign is_ptr1 = &(pc_ptr1[10:7] ^~ mem_id);
17
                                                                                                            17
                                                                                                                 assign is_ptr2 = &(pc_ptr2[10:7] ^~ mem_id);
                                                                                                            18
18
                                                                                                                 assign sel ptr = {is ptr2,is ptr1,is ptr0};
            pc_ptr0 <= SQ_SX_ptr0;
20
           pc_ptr1 \le SQ_SX_ptr1;
                                                                                                            20 reg [6:0] pc_index; //read pointer from the Parameter caches
            pc_ptr2 <= SQ_SX_ptr2;
21
                                                                                                            21
                                                                                                                  reg [0:0] pc rd en;
            pc_wr_en <= SQ_SX_pc_wr_en;
             pc_wr_addr <= SQ_SX_pc_wr_addr;
                                                                                                            23
                                                                                                                  always@(is_ptr0 or is_ptr1 or is_ptr2 or pc_ptr0 or pc_ptr1 or pc_ptr2 or sel_ptr)
             pc_cmask <= SQ_SX_pc_cmask;
                                                                                                            24
             vertex_data_in <= SP_SX_data;
                                                                                                                       case(sel ptr)
                                       Page 3 of 14
                                                                                                                                                   Page 4 of 14
                                                           Ex. 2111 - param cache ctl.v
                                                                                                                                                                        Ex. 2111 - param cache ctl.v
```

ATI 2111 LG v. ATI IPR2015-00326

```
3'b000-
                                                                                                                                                                                      endcase // case(sel ptr)
 2
                    begin
                                                                                                                                                                                end // always@ (is_ptr0 or is_ptr1 or is_ptr2 or pc_ptr0 or pc_ptr1 or pc_ptr2)
                      pc_index = pc_ptr0;
                      pc_rd_en = 1'b0;
                                                                                                                                                                             `ifdef USE_BEHAVE_MEM
                                                                                                                                                                               dum_mem_p2 #(7,128) u_pc(.iRCLK(sclk),
                   3'b001:
                                                                                                                                                                                                           .iWCLK(sclk),
                      pc_index = pc_ptr0;
                                                                                                                                                                                                            .iMER(pc_rd_en),
                     pc_rd_en = 1'b1;
                                                                                                                                                                                                            .iMEW(pc_wr_en),
10
                                                                                                                                                                                                            .iWEN(pc_wr_en),
11
                  3'b010:
                                                                                                                                                                                                            .iRADR(pc_index),
12
                                                                                                                                                                                                           .iWADR(pc_wr_addr),
                    begin
13
                      pc index = pc ptrl;
                                                                                                                                                                                                            .iD(vertex_data_in),
14
                      pc rd en = 1'b1;
                                                                                                                                                                                                            .oQ(vertex data out)
15
                                                                                                                                                                    15
                    end
                                                                                                                                                                                                           );
                                                                                                                                                                           'else // !'ifdef USE BEHAVE MEM
16
                  3'b100:
                                                                                                                                                                     16
17
                                                                                                                                                                    17
                    begin
                                                                                                                                                                              rfsd2 128x128cm2sw0 u pc
18
                      pc_index = pc_ptr2;
                                                                                                                                                                     18
                                                                                                                                                                               /\!/ tp\_coord\_fifo\_ram\ utp\_coord\_fifo\_ram\_0
19
                      pc_rd_en = 1'b1;
                                                                                                                                                                           // (/*VRGIO tp_coord_fifo_ram cfifo_in cfifo_out q_cfifo_wptr q_cfifo_rptr cfifo_ram_wen cfifo_ram_ren 10^*/
                                                                                                                                                                     20
20
                    end
                                                                                                                                                                    21
21
                  default:
                                                                                                                                                                                (/*VRGIO rfsd2_128x128cm2sw0 vertex_data_in vertex_data_out pc_wr_addr pc_index
22
                                                                                                                                                                            pc_wr_en pc_rd_en null*
23
                      pc_index = pc_ptr0;
                                                                                                                                                                    24
24
                      pc_rd_en = 1'b0;
                                                                                                                                                                                 .CLKB(iSCLK), // Read Clock
                                                                                                                                                                                  .OEB(pc_rd_en), // Output enable
                                                                                                                                                                                                                                  Page 6 of 14
                                                             Page 5 of 14
                                                                                           Ex. 2111 - param_cache_ctl.v
                                                                                                                                                                                                                                                               Ex. 2111 - param_cache_ctl.v
                                                                                                                                                                            . QB56(vertex_data_out[56]), . QB57(vertex_data_out[57]), . . QB58(vertex_data_out[58]), . QB59(vertex_data_out[59]), \# (Read Data
             .MEB(vdd), // Read enable
        . ADRB0(pc\_index[0]), \\ . ADRB1(pc\_index[1]), \\ . ADRB3(pc\_index[3]), \ /\!/ \ Read \ Address
                                                                                                   .ADRB2(pc index[2]),
                                                                                                                                                                             . QB60(vertex\_data\_out[60]), \quad QB61(vertex\_data\_out[61]), \quad . QB62(vertex\_data\_out[62]), \\ . QB63(vertex\_data\_out[63]), \\ // Read Data
             .ADRB4(pc_index[4]), .ADRB5(pc_index[5]), .ADRB6(pc_index[6]), // Read Address
                                                                                                                                                                             .QB64(vertex_data_out[64]), .QB65(vertex_data_out[65]), .QB66(vertex_data_out[66]), .QB67(vertex_data_out[67]), // Read Data
        . QB0(vertex\_data\_out[0]), \qquad . QB \\ . QB3(vertex\_data\_out[3]), \ // \ Read \ Data
                                                     . QB1(vertex\_data\_out[1]),
                                                                                               .QB2(vertex_data_out[2]),
                                                                                                                                                                             .QB68(vertex\_data\_out[68]), \quad .QB69(vertex\_data\_out[69]), \quad .QB70(vertex\_data\_out[70]), \\ .QB71(vertex\_data\_out[71]), \\ // Read Data
        .QB4(vertex_data_out[4]), .QB
.QB7(vertex_data_out[7]), // Read Data
                                                      .QB5(vertex_data_out[5]),
                                                                                               .QB6(vertex_data_out[6]),
                                                                                                                                                                             .QB72(vertex_data_out[72]), .QB73(vertex_data_out[73]), .QB74(vertex_data_out[74]), .QB75(vertex_data_out[75]), // Read Data
        . QB8(vertex\_data\_out[8]), \qquad . QB9(vertex\_data\_out[9]), \qquad . QB10(vertex\_data\_out[10]), \\ . QB11(vertex\_data\_out[11]), \ // \ Read \ Data
                                                                                                                                                                             .QB76(vertex_data_out[76]), .QB77(vertex_data_out[77]), .QB78(vertex_data_out[78]), .QB79(vertex_data_out[79]), // Read Data
        . QB12(vertex_data_out[12]), . QB13(vertex_data_out[13]), .QB14(vertex_data_out[14]), .QB15(vertex_data_out[15]), // Read Data
11
12
                                                                                                                                                                             . QB80(vertex\_data\_out[80]), \quad . QB81(vertex\_data\_out[81]), \quad . QB82(vertex\_data\_out[82]), \\ . QB83(vertex\_data\_out[83]), \\ // Read Data
        . QB16(vertex\_data\_out[16]), \quad . QB17(vertex\_data\_out[17]), \quad . QB18(vertex\_data\_out[18]), \\ . QB19(vertex\_data\_out[19]), \quad // Read \ Data
13
14
                                                                                                                                                                             . QB84(vertex\_data\_out[84]), \quad . QB85(vertex\_data\_out[85]), \quad . QB86(vertex\_data\_out[86]), \\ . QB87(vertex\_data\_out[87]), \quad // Read Data
        .QB20(vertex_data_out[20]), .QB21(vertex_data_out[21]), .QB22(vertex_data_out[22]), .QB23(vertex_data_out[23]), // Read Data
                                                                                                                                                                             .QB88(vertex_data_out[88]), .QB89(vertex_data_out[89]), .QB90(vertex_data_out[90]), .QB91(vertex_data_out[91]), // Read Data
        .QB24(vertex_data_out[24]), .QB25(vertex_data_out[25]), .QB26(vertex_data_out[26]), .QB27(vertex_data_out[27]), // Read Data
17
18
                                                                                                                                                                            . QB92(vertex\_data\_out[92]), \quad . QB93(vertex\_data\_out[93]), \quad . QB94(vertex\_data\_out[94]), \\ . QB95(vertex\_data\_out[95]), \\ // Read Data
        .QB28(vertex_data_out[28]), .QB29(vertex_data_out[29]), .QB30(vertex_data_out[30]), .QB31(vertex_data_out[31]), // Read Data
20
                                                                                                                                                                             . QB96(vertex\_data\_out[96]), \quad . QB97(vertex\_data\_out[97]), \quad . QB98(vertex\_data\_out[98]), \\ . QB99(vertex\_data\_out[99]), \\ // Read \ Data
        . QB32(vertex_data_out[32]), . QB33(vertex_data_out[33]), . QB34(vertex_data_out[34]), . QB35(vertex_data_out[35]), . // Read Data
21
22
                                                                                                                                                                                  .QB100(vertex_data_out[100]),
                                                                                                                                                                                                                                                              .QB101(vertex_data_out[101]),
23
24
        .QB36(vertex_data_out[36]), .QB37(vertex_data_out[37]), .QB38(vertex_data_out[38]), .QB39(vertex_data_out[39]), // Read Data
                                                                                                                                                                             .QB102(vertex_data_out[102]), .QB103(vertex_data_out[103]), // Read Data
                                                                                                                                                                             .QB104(vertex_data_out[104]), .QB105(vertex_data_out[105]), .QB106(vertex_data_out[106]), .QB107(vertex_data_out[107]), // Read Data
        .QB40(vertex_data_out[40]), .QB41(vertex_data_out[41]), .QB42(vertex_data_out[42]), .QB43(vertex_data_out[43]), // Read Data
                                                                                                                                                                             . QB108(vertex\_data\_out[108]), \\ . QB109(vertex\_data\_out[109]), \\ . QB10(vertex\_data\_out[110]), . QB111(vertex\_data\_out[111]), \\ // Read\ Data
        .QB44(vertex_data_out[44]), .QB45(vertex_data_out[45]), .QB46(vertex_data_out[46]), .QB47(vertex_data_out[47]), // Read Data
                                                                                                                                                                             . QB112(vertex\_data\_out[112]), \qquad . QB113(vertex\_data\_out[113]), \\ . QB114(vertex\_data\_out[114]), . QB115(vertex\_data\_out[115]), \\ // Read Data
        .QB48(vertex_data_out[48]), .QB49(vertex_data_out[49]), .QB50(vertex_data_out[50]), .QB51(vertex_data_out[51]), // Read Data
                                                                                                                                                                             . QB116(vertex\_data\_out[116]), \\ . QB118(vertex\_data\_out[117]), \\ . QB18(vertex\_data\_out[118]), \\ . QB19(vertex\_data\_out[119]), \\ // Read \ Data
        . QB52(vertex\_data\_out[52]), \quad . QB53(vertex\_data\_out[53]), \quad . QB54(vertex\_data\_out[54]), \\ . QB55(vertex\_data\_out[55]), \ // Read \ Data
31
32
                                                            Page 7 of 14
                                                                                                                                                                                                                                  Page 8 of 14
                                                                                          Ex. 2111 - param cache ctl.v
                                                                                                                                                                                                                                                               Ex. 2111 - param cache ctl.v
```

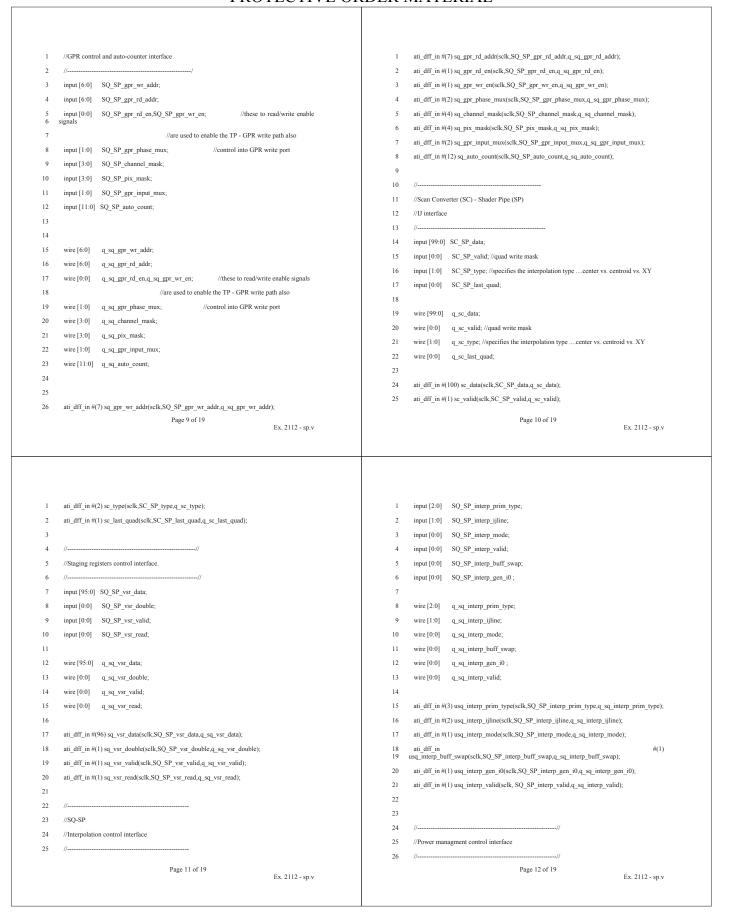
1 2	.QB120(vertex_data_out[120]), .QB121(vertex_data_out[121]), .QB122(vertex_data_out[121]), .QB122(vertex_data_out[122]), .QB123(vertex_data_out[123]), .// Read_Data	1 .DA40(vertex_data_in[40]), .DA41(vertex_data_in[41]), .DA42(vertex_data_in[42]), 2 .DA43(vertex_data_in[43]), // Write Data
3 4	.QB124(vertex_data_out[124]), .QB126(vertex_data_out[127]), ./Read Jata .QB124(vertex_data_out[124]), .QB126(vertex_data_out[127]), ./Read Jata	3DA44(vertex_data_in[44]),DA45(vertex_data_in[45]),DA46(vertex_data_in[46]), 4DA47(vertex_data_in[47]), // Write Data
5	// WRITE INTERFACE	5 .DA48(vertex data in[48]), .DA49(vertex data in[49]), .DA50(vertex data in[50]),
6	.CLKA(iSCLK), // Write Clock	6 .DA51(vertex_data_in[51]),
7	.WEA(pc_wr_en), // Write enable .MEA(vdd), // Memory enable	8 .DA55(vertex_data_in[55]), // Write Data 9 .DA56(vertex_data_in[56]), .DA57(vertex_data_in[57]), .DA58(vertex_data_in[58]),
9 10	.ADRA0(pc_wr_addr[0]), .ADRA1(pc_wr_addr[1]), .ADRA2(pc_wr_addr[2]), .ADRA3(pc_wr_addr[3]), // Write Address	10 .DA59(vertex_data_in[59]), // Write Data 11 .DA60(vertex_data_in[60]), .DA61(vertex_data_in[61]), .DA62(vertex_data_in[62]),
11 12	.ADRA4(pc_wr_addr[4]), .ADRA5(pc_wr_addr[5]), .ADRA6(pc_wr_addr[6]), // Write Address	12 .DA63(vertex_data_in[63]), // Write Data 13 .DA64(vertex_data_in[64]), .DA65(vertex_data_in[65]), .DA66(vertex_data_in[66]),
13 14	.DA0(vertex_data_in[0]), .DA1(vertex_data_in[1]), .DA2(vertex_data_in[2]), .DA3(vertex_data_in[3]), // Write Data	14 .DA67(vertex_data_in[67]), // Write Data
15	.DA4(vertex_data_in[4]), .DA5(vertex_data_in[5]), .DA6(vertex_data_in[6]),	16 .DA71(vertex_data_in[71]), // Write Data
16 17	.DA7(vertex_data_in[7]), // Write Data .DA8(vertex_data_in[8]), .DA9(vertex_data_in[9]), .DA10(vertex_data_in[10]),	17 .DA72(vertex_data_in[72]), .DA73(vertex_data_in[73]), .DA74(vertex_data_in[74]), 18 .DA75(vertex_data_in[75]), // Write Data
18 19	.DA12(vertex_data_in[11]), // Write Data .DA12(vertex_data_in[12]), .DA13(vertex_data_in[13]), .DA14(vertex_data_in[14]),	19 .DA76(vertex_data_in[76]), .DA77(vertex_data_in[77]), .DA78(vertex_data_in[78]), 20 .DA79(vertex_data_in[79]), // Write Data
20 21	.DA15(vertex_data_in[15]), // Write Data .DA16(vertex_data_in[16]), .DA17(vertex_data_in[17]), .DA18(vertex_data_in[18]),	21 .DA80(vertex_data_in[80]), .DA81(vertex_data_in[81]), .DA82(vertex_data_in[82]), 22 .DA83(vertex_data_in[83]), // Write Data
22	DA20(vertex_data_in[20]),DA21(vertex_data_in[21]),DA22(vertex_data_in[22]),DA21(vertex_data_in[22]),DA21(vertex_data_in[21]),DA22(vertex_data_in[22]),DA21(vertex_data_in[21]),DA21(vertex_data	23 .DA84(vertex_data_in[84]), .DA85(vertex_data_in[85]), .DA86(vertex_data_in[86]), 24 .DA87(vertex_data_in[87]), // Write Data
24	.DA23(vertex_data_in[23]), // Write Data	25 .DA88(vertex_data_in[88]), .DA89(vertex_data_in[89]), .DA90(vertex_data_in[90]), 26 .DA91(vertex_data_in[91]), // Write Data
25 26	.DA27(vertex_data_in[27]), // Write Data	27 .DA92(vertex_data_in[92]), .DA93(vertex_data_in[93]), .DA94(vertex_data_in[94]), 28 .DA95(vertex_data_in[95]), // Write Data
27 28	.DA28(vertex_data_in[28]), .DA29(vertex_data_in[29]), .DA30(vertex_data_in[30]), .DA31(vertex_data_in[31]), // Write Data	29 .DA96(vertex_data_in[96]), .DA97(vertex_data_in[97]), .DA98(vertex_data_in[98]), 30 .DA99(vertex_data_in[99]), // Write Data
29 30	.DA32(vertex_data_in[32]), .DA33(vertex_data_in[33]), .DA34(vertex_data_in[34]), .DA35(vertex_data_in[35]), // Write Data	31 .DA100(vertex_data_in[101]), .DA101(vertex_data_in[101]), .DA102(vertex_data_in[102]), .DA103(vertex_data_in[103]), // Write Data
31 32	.DA36(vertex_data_in[36]), .DA37(vertex_data_in[37]), .DA38(vertex_data_in[38]), .DA39(vertex_data_in[39]), // Write Data	
	Page 9 of 14 Ex. 2111 - param_cache_ctl.v	Page 10 of 14 Ex. 2111 - param_cache_ctl.v
2	.DA104(vertex_data_in[104]), .DA105(vertex_data_in[105]), .DA106(vertex_data_in[106]), .DA107(vertex_data_in[107]), // Write Data	1TDA24(vertex_data_in[24]),TDA25(vertex_data_in[25]),TDA26(vertex_data_in[26]), 2TDA27(vertex_data_in[27]), // Write Test Data
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22		2 .TDA27(vertex_data_in[27]), // Write Test Data 3 .TDA28(vertex_data_in[28]), .TDA29(vertex_data_in[29]), .TDA30(vertex_data_in[30]), 4 .TDA31(vertex_data_in[31]), // Write Test Data 5 .TDA32(vertex_data_in[32]), .TDA33(vertex_data_in[33]), .TDA34(vertex_data_in[34]), 6 .TDA35(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[36]), .TDA37(vertex_data_in[37]), .TDA38(vertex_data_in[38]), 8 .TDA39(vertex_data_in[39]), // Write Test Data 9 .TDA40(vertex_data_in[40]), .TDA41(vertex_data_in[41]), .TDA42(vertex_data_in[42]), 10 .TDA43(vertex_data_in[43]), // Write Test Data 11 .TDA44(vertex_data_in[47]), // Write Test Data 12 .TDA44(vertex_data_in[47]), // Write Test Data 13 .TDA48(vertex_data_in[47]), // Write Test Data 14 .TDA54(vertex_data_in[47]), // Write Test Data 15 .TDA52(vertex_data_in[51]), // Write Test Data 16 .TDA52(vertex_data_in[51]), // Write Test Data 17 .TDA55(vertex_data_in[51]), // Write Test Data 18 .TDA55(vertex_data_in[52]), .TDA53(vertex_data_in[53]), .TDA54(vertex_data_in[54]), .TDA55(vertex_data_in[59]), // Write Test Data 19 .TDA65(vertex_data_in[59]), // Write Test Data 19 .TDA66(vertex_data_in[63]), // Write Test Data 20 .TDA64(vertex_data_in[63]), // Write Test Data 21 .TDA64(vertex_data_in[63]), // Write Test Data 22 .TDA66(vertex_data_in[63]), // Write Test Data 23 .TDA66(vertex_data_in[64]), .TDA65(vertex_data_in[65]), .TDA66(vertex_data_in[66]), .TDA66(vertex_data_
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	DA108(vertex_data_in[108]), DA107(vertex_data_in[107]), // Write Data DA108(vertex_data_in[108]), DA111(vertex_data_in[111]), // Write Data DA112(vertex_data_in[112]), DA111(vertex_data_in[111]), // Write Data DA112(vertex_data_in[112]), DA113(vertex_data_in[113]), // Write Data DA118(vertex_data_in[114]), DA115(vertex_data_in[115]), // Write Data DA118(vertex_data_in[118]), DA119(vertex_data_in[119]), // Write Data DA120(vertex_data_in[120]), DA123(vertex_data_in[123]), // Write Data DA122(vertex_data_in[120]), DA123(vertex_data_in[123]), // Write Data DA124(vertex_data_in[124]), DA126(vertex_data_in[125]), // Write Data DA125(vertex_data_in[126]), DA127(vertex_data_in[127]), // Write Data // WRITE TEST SIGNALS BISTEA(vss), TWEA(vss), TADRA0(pc_wr_addr[0]), TADRA1(pc_wr_addr[1]), TADRA2(pc_wr_addr[2]), TADRA4(pc_wr_addr[4]), TADRA4(pc_wr_addr[4]), TADRA6(pc_wr_addr[6]), // Write Test Address TDA0(vertex_data_in[0]), TDA1(vertex_data_in[0]), TDA2(vertex_data_in[0]), TAD2(vertex_data_in[0]), T	2 .TDA2{(vertex_data_in[27]), // Write Test Data 3 .TDA28(vertex_data_in[28]), .TDA29(vertex_data_in[29]), .TDA30(vertex_data_in[30]), 4 .TDA31(vertex_data_in[31]), // Write Test Data 5 .TDA32(vertex_data_in[32]), .TDA33(vertex_data_in[33]), .TDA34(vertex_data_in[34]), 6 .TDA35(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[35]), // Write Test Data 8 .TDA39(vertex_data_in[39]), // Write Test Data 9 .TDA40(vertex_data_in[40]), .TDA41(vertex_data_in[41]), .TDA42(vertex_data_in[42]), 10 .TDA43(vertex_data_in[44]), .TDA45(vertex_data_in[45]), .TDA46(vertex_data_in[46]), 11 .TDA43(vertex_data_in[44]), .TDA45(vertex_data_in[45]), .TDA46(vertex_data_in[46]), 12 .TDA47(vertex_data_in[47]), // Write Test Data 13 .TDA48(vertex_data_in[48]), .TDA49(vertex_data_in[49]), .TDA50(vertex_data_in[50]), 14 .TDA51(vertex_data_in[51]), // Write Test Data 15 .TDA52(vertex_data_in[55]), .TDA53(vertex_data_in[53]), .TDA54(vertex_data_in[54]), 16 .TDA56(vertex_data_in[55]), .TDA57(vertex_data_in[57]), .TDA58(vertex_data_in[58]), 17 .TDA50(vertex_data_in[59]), .TDA57(vertex_data_in[57]), .TDA58(vertex_data_in[58]), 18 .TDA50(vertex_data_in[66]), .TDA61(vertex_data_in[61]), .TDA62(vertex_data_in[62]), 19 .TDA63(vertex_data_in[63]), .// Write Test Data 21 .TDA64(vertex_data_in[63]), .// Write Test Data 22 .TDA68(vertex_data_in[63]), .// Write Test Data 23 .TDA68(vertex_data_in[67]), .// Write Test Data 24 .TDA71(vertex_data_in[67]), .// Write Test Data 25 .TDA68(vertex_data_in[67]), .// Write Test Data 26 .TDA71(vertex_data_in[67]), .// Write Test Data 27 .TDA67(vertex_data_in[67]), .// Write Test Data 28 .TDA68(vertex_data_in[67]), .// Write Test Data 29 .TDA68(vertex_data_in[67]), .// Write Test Data 20 .TDA68(vertex_data_in[67]), .// Write Test Data 21 .TDA68(vertex_data_in[67]), .// Write Test Data 22 .TDA68(vertex_data_in[67]), .// Write Test Data 23 .TDA68(vertex_data_in[67]), .// Write Test Data 24 .TDA71(vertex_data_in[67]), .// Write Test Data 25 .TDA68(vertex_data_in[67]), .// Write Test Data 26 .TDA71(vertex_d
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	DA108(vertex_data_in[108]), DA107(vertex_data_in[107]), // Write Data DA108(vertex_data_in[108]), DA111(vertex_data_in[111]), // Write Data DA112(vertex_data_in[112]), DA1112(vertex_data_in[111]), // Write Data DA112(vertex_data_in[114]), DA115(vertex_data_in[115]), // Write Data DA116(vertex_data_in[116]), DA117(vertex_data_in[117]), // Write Data DA120(vertex_data_in[120]), DA121(vertex_data_in[121]), // Write Data DA120(vertex_data_in[120]), DA123(vertex_data_in[123]), // Write Data DA120(vertex_data_in[124]), DA123(vertex_data_in[123]), // Write Data DA124(vertex_data_in[124]), DA125(vertex_data_in[127]), // Write Data // WRITE TEST SIGNALS BISTEA(vss), TWEA(vss), TADRA0(pc_wr_addr[0]), .TADRA1(pc_wr_addr[1]), .TADRA2(pc_wr_addr[2]), .TADRA3(pc_wr_addr[3]), // Write Test Address TDA0(vertex_data_in[0]), .TDA1(vertex_data_in[1]), .TDA2(vertex_data_in[2]), .TDA3(vertex_data_in[0]), .TDA1(vertex_data_in[1]), .TDA2(vertex_data_in[2]), .TDA3(vertex_data_in[3]), // Write Test Data TDA4(vertex_data_in[4]), .TDA5(vertex_data_in[5]), .TDA6(vertex_data_in[6]), .TDA5(vertex_data_in[6]),	2 .TDA27(vertex_data_in[27]), // Write Test Data 3 .TDA28(vertex_data_in[28]), .TDA29(vertex_data_in[29]), .TDA30(vertex_data_in[30]), 4 .TDA31(vertex_data_in[31]), // Write Test Data 5 .TDA32(vertex_data_in[35]), .TDA33(vertex_data_in[33]), .TDA34(vertex_data_in[34]), 6 .TDA35(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[36]), .TDA37(vertex_data_in[37]), .TDA38(vertex_data_in[38]), 8 .TDA39(vertex_data_in[39]), // Write Test Data 9 .TDA40(vertex_data_in[40]), .TDA41(vertex_data_in[41]), .TDA42(vertex_data_in[42]), 10 .TDA43(vertex_data_in[43]), // Write Test Data 11 .TDA44(vertex_data_in[47]), // Write Test Data 12 .TDA48(vertex_data_in[47]), // Write Test Data 13 .TDA48(vertex_data_in[47]), // Write Test Data 14 .TDA54(vertex_data_in[51]), // Write Test Data 15 .TDA52(vertex_data_in[51]), // Write Test Data 16 .TDA55(vertex_data_in[55]), // Write Test Data 17 .TDA55(vertex_data_in[55]), // Write Test Data 18 .TDA55(vertex_data_in[55]), // Write Test Data 19 .TDA56(vertex_data_in[56]), .TDA57(vertex_data_in[57]), .TDA58(vertex_data_in[58]), .TDA59(vertex_data_in[58]), .TDA59(vertex_data_in[58]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[67]), // Write Test Data 20 .TDA60(vertex_data_in[67]), // Write Test Data 21 .TDA60(vertex_data_in[67]), // Write Test Data 22 .TDA68(vertex_data_in[67]), // Write Test Data 23 .TDA68(vertex_data_in[67]), // Write Test Data 24 .TDA66(vertex_data_in[67]), // Write Test Data 25 .TDA68(vertex_data_in[67]), // Write Test Data 26 .TDA67(vertex_data_in[67]), // Write Test Data 27 .TDA76(vertex_data_in[77]), // Write Test Data 28 .TDA68(vertex_data_in[67]), // Write Test Data 29 .TDA68(vertex_data_in[67]), // Write Test Data 29 .TDA68(vertex_data_in[67]), // Write Test Data 20 .TDA67(vertex_data_in[67]), // Write Test Data 21 .TDA66(vertex_data_in[67]), // Write Test Data 22 .TDA68(vertex_data_in[67]), // Write Test Data 23 .TDA68(vertex_data_in[67]), // Write Test Data 24 .TDA68(vertex_data_in[67]), /
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24		2 .TDA2f(vertex_data_in[27]), // Write Test Data 3 .TDA28(vertex_data_in[28]), .TDA29(vertex_data_in[29]), .TDA30(vertex_data_in[30]), 4 .TDA31(vertex_data_in[31]), // Write Test Data 5 .TDA32(vertex_data_in[32]), .TDA33(vertex_data_in[33]), .TDA34(vertex_data_in[34]), 6 .TDA35(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[39]), // Write Test Data 8 .TDA36(vertex_data_in[39]), // Write Test Data 9 .TDA40(vertex_data_in[40]), .TDA41(vertex_data_in[41]), .TDA42(vertex_data_in[42]), 10 .TDA43(vertex_data_in[44]), .TDA45(vertex_data_in[45]), .TDA46(vertex_data_in[46]), 11 .TDA43(vertex_data_in[44]), .TDA45(vertex_data_in[45]), .TDA46(vertex_data_in[46]), 12 .TDA47(vertex_data_in[47]), // Write Test Data 13 .TDA48(vertex_data_in[48]), .TDA49(vertex_data_in[49]), .TDA50(vertex_data_in[50]), 14 .TDA51(vertex_data_in[51]), // Write Test Data 15 .TDA52(vertex_data_in[52]), .TDA53(vertex_data_in[53]), .TDA54(vertex_data_in[54]), 16 .TDA55(vertex_data_in[55]), // Write Test Data 17 .TDA56(vertex_data_in[59]), // Write Test Data 18 .TDA66(vertex_data_in[59]), // Write Test Data 19 .TDA60(vertex_data_in[60]), .TDA61(vertex_data_in[61]), .TDA62(vertex_data_in[62]), 11 .TDA64(vertex_data_in[63]), // Write Test Data 21 .TDA64(vertex_data_in[63]), // Write Test Data 22 .TDA66(vertex_data_in[61]), .TDA66(vertex_data_in[62]), .TDA66(vertex_data_in[61]), .TDA66(vertex_data_in[61]), .TDA67(vertex_data_in[61]), .TDA66(vertex_data_in[61]), .TDA67(vertex_data_in[61]), .TDA67(vertex_data_in[61]), .TDA67(vertex_data_in[61]), .TDA67(vertex_data_in[61]), .TDA67(vertex_data_in[61]), .TDA67(vertex_data_in[62]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	DA108(vertex_data_in[108]), DA107(vertex_data_in[107]), // Write Data DA108(vertex_data_in[108]), DA111(vertex_data_in[111]), // Write Data DA112(vertex_data_in[112]), DA115(vertex_data_in[111]), // Write Data DA112(vertex_data_in[112]), DA115(vertex_data_in[115]), // Write Data DA116(vertex_data_in[114]), DA115(vertex_data_in[115]), // Write Data DA117(vertex_data_in[118]), DA119(vertex_data_in[119]), // Write Data DA120(vertex_data_in[120]), DA123(vertex_data_in[123]), // Write Data DA120(vertex_data_in[120]), DA123(vertex_data_in[123]), // Write Data DA124(vertex_data_in[124]), DA125(vertex_data_in[123]), // Write Data DA126(vertex_data_in[124]), DA127(vertex_data_in[127]), // Write Data DA126(vertex_data_in[126]), DA127(vertex_data_in[127]), // Write Data // WRITE TEST SIGNALS BISTEA(vss), .TWEA(vss), .TADRA0(pc_wr_addr[0]), .TADRA1(pc_wr_addr[1]), .TADRA2(pc_wr_addr[2]), .TADRA3(pc_wr_addr[3]), // Write Test Address .TDA0(vertex_data_in[4]), .TADRA5(pc_wr_addr[5]), .TADRA6(pc_wr_addr[6]), // Write Test Address .TDA4(vertex_data_in[4]),TDA5(vertex_data_in[5]), .TDA2(vertex_data_in[6]), .TDA3(vertex_data_in[7]), // Write Test Data .TDA4(vertex_data_in[7]), // Write Test Data .TDA8(vertex_data_in[7]), // Write Test Data .TDA8(vertex_data_in[1]), // Write Test Data .TDA9(vertex_data_in[1]), // Write Test Data .TDA12(vertex_data_in[1]), // Write Test Data	2 .TDA27(vertex_data_in[27]), // Write Test Data 3 .TDA38(vertex_data_in[31]), // Write Test Data 4 .TDA31(vertex_data_in[31]), // Write Test Data 5 .TDA32(vertex_data_in[31]), // Write Test Data 6 .TDA35(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[36]), // Write Test Data 8 .TDA39(vertex_data_in[36]), // Write Test Data 9 .TDA40(vertex_data_in[40]), // Write Test Data 11 .TDA44(vertex_data_in[43]), // Write Test Data 12 .TDA44(vertex_data_in[43]), // Write Test Data 13 .TDA44(vertex_data_in[47]), // Write Test Data 14 .TDA47(vertex_data_in[47]), // Write Test Data 15 .TDA48(vertex_data_in[47]), // Write Test Data 16 .TDA52(vertex_data_in[47]), // Write Test Data 17 .TDA51(vertex_data_in[51]), // Write Test Data 18 .TDA52(vertex_data_in[55]), // Write Test Data 19 .TDA55(vertex_data_in[55]), // Write Test Data 19 .TDA55(vertex_data_in[55]), // Write Test Data 19 .TDA56(vertex_data_in[56]), .TDA57(vertex_data_in[57]), .TDA58(vertex_data_in[58]), .TDA58(vertex_data_in[58]), .TDA59(vertex_data_in[58]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[56]), .TDA56(vertex_data_in[67]), // Write Test Data 19 .TDA60(vertex_data_in[66]), .TDA61(vertex_data_in[65]), .TDA66(vertex_data_in[66]), .TDA66(vertex_data_in[67]), // Write Test Data 20 .TDA64(vertex_data_in[67]), // Write Test Data 21 .TDA66(vertex_data_in[67]), // Write Test Data 22 .TDA68(vertex_data_in[67]), // Write Test Data 23 .TDA68(vertex_data_in[67]), // Write Test Data 24 .TDA66(vertex_data_in[67]), // Write Test Data 25 .TDA66(vertex_data_in[67]), // Write Test Data 26 .TDA67(vertex_data_in[67]), // Write Test Data 27 .TDA76(vertex_data_in[67]), // Write Test Data 28 .TDA76(vertex_data_in[67]), // Write Test Data 29 .TDA66(vertex_data_in[67]), // Write Test Data 29 .TDA66(vertex_da
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	DA108(vertex_data_in[108]), DA107(vertex_data_in[107]), // Write Data DA108(vertex_data_in[108]), DA111(vertex_data_in[111]), // Write Data DA112(vertex_data_in[112]), DA115(vertex_data_in[115]), // Write Data DA112(vertex_data_in[114]), DA115(vertex_data_in[115]), // Write Data DA116(vertex_data_in[116]), DA115(vertex_data_in[115]), // Write Data DA116(vertex_data_in[118]), DA119(vertex_data_in[119]), // Write Data DA110(vertex_data_in[120]), DA121(vertex_data_in[117]), DA122(vertex_data_in[121]), DA122(vertex_data_in[122]), DA123(vertex_data_in[122]), DA123(vertex_data_in[122]), DA124(vertex_data_in[122]), DA125(vertex_data_in[125]), DA126(vertex_data_in[126]), DA127(vertex_data_in[127]), // Write Data DA124(vertex_data_in[126]), DA127(vertex_data_in[127]), // Write Data DA125(vertex_data_in[126]), DA127(vertex_data_in[127]), // Write Data // WRITE TEST SIGNALS BISTEA(vss), TWEA(vss), TMEA(vss), TADRA3(pc_wr_addr[0]), .TADRA1(pc_wr_addr[1]), .TADRA2(pc_wr_addr[2]), .TADRA3(pc_wr_addr[3]), // Write Test Address TDA0(vertex_data_in[3]), // Write Test Data TDA4(vertex_data_in[3]), // Write Test Data TDA4(vertex_data_in[4]), .TDA5(vertex_data_in[5]), .TDA6(vertex_data_in[6]), .TDA7(vertex_data_in[1]), .TDA12(vertex_data_in[1]), .TDA13(vertex_data_in[1]), .TDA14(vertex_data_in[1]), .TDA14(vertex_data_in[1]), .TDA15(vertex_data_in[1]), .TDA16(vertex_data_in[1]), .	2 .TDA28(vertex_data_in[27]), // Write Test Data 3 .TDA38(vertex_data_in[31]), // Write Test Data 4 .TDA31(vertex_data_in[31]), // Write Test Data 5 .TDA32(vertex_data_in[32]), .TDA33(vertex_data_in[33]), .TDA34(vertex_data_in[34]), .TDA35(vertex_data_in[35]), // Write Test Data 7 .TDA36(vertex_data_in[35]), // Write Test Data 8 .TDA39(vertex_data_in[39]), // Write Test Data 9 .TDA40(vertex_data_in[40]), .TDA41(vertex_data_in[41]), .TDA42(vertex_data_in[42]), .TDA43(vertex_data_in[43]), // Write Test Data 11 .TDA44(vertex_data_in[44]), .TDA45(vertex_data_in[45]), .TDA46(vertex_data_in[46]), .TDA47(vertex_data_in[47]), // Write Test Data 12 .TDA48(vertex_data_in[44]), .TDA49(vertex_data_in[49]), .TDA50(vertex_data_in[50]), .TDA51(vertex_data_in[51]), // Write Test Data 13 .TDA52(vertex_data_in[51]), // Write Test Data 14 .TDA51(vertex_data_in[52]), .TDA53(vertex_data_in[53]), .TDA54(vertex_data_in[54]), .TDA55(vertex_data_in[55]), .TDA55(vertex_data_in[55]), .TDA56(vertex_data_in[59]), .TDA56(vertex_data_in[60]), .TDA66(vertex_data_in[60]), .TDA66(ver

```
. TDA88(vertex_data_in[88]), . TDA89(vertex_data_in[89]), . TDA90(vertex_data_in[90]), . TDA91(vertex_data_in[91]), // Write Test Data
                                                                                                                                                                  1
                                                                                                                                                                 2
                                                                                                                                                                         //assign the read values to the out port
        .
TDA92(vertex_data_in[92]), .
TDA95(vertex_data_in[93]), .
TDA95(vertex_data_in[95]), // Write Test Data
                                                                                                                                                                         assign ovtx_data = q0_vertex_data_out; // the output of the parameter cache is registered...???????
        .
TDA96(vertex_data_in[96]), .
TDA97(vertex_data_in[97]), .
TDA98(vertex_data_in[98]), .
TDA99(vertex_data_in[99]), // Write Test Data
                                                                                                                                                                 5
        . TDA100(vertex\_data\_in[100]), \qquad . TDA101(vertex\_data\_in[101]), \\. TDA102(vertex\_data\_in[\overline{102}]), . TDA103(vertex\_data\_in[103]), \\ // Write Test Data
                                                                                                                                                                 6
                                                                                                                                                                       endmodule // param cache ctl
             .TDA104(vertex_data_in[104]),
                                                                                     .TDA105(vertex_data_in[105]),
        .TDA106(vertex_data_in[106]), .TDA107(vertex_data_in[107]), // Write Test Data
10
        . TDA108(vertex\_data\_in[108]), \qquad . TDA109(vertex\_data\_in[109]), \\ . TDA110(vertex\_data\_in[110]), . TDA111(vertex\_data\_in[111]), \\ // Write Test Data
12
                                                                                                                                                                 10
        . TDA112(vertex_data_in[112]), . TDA113(vertex_data_in[113]), . TDA114(vertex_data_in[114]), . TDA115(vertex_data_in[115]), // Write Test Data
                                                                                                                                                                 11
                                                                                                                                                                 12
             .TDA116(vertex data in[116]),
                                                                                     .TDA117(vertex_data_in[117]),
        .TDA118(vertex_data_in[118]), .TDA119(vertex_data_in[119]), // Write Test Data
                                                                                                                                                                 13
        . TDA120 (vertex\_data\_in[120]), \\ . TDA121 (vertex\_data\_in[121]), \\ . TDA122 (vertex\_data\_in[122]), . TDA123 (vertex\_data\_in[123]), \\ // Write Test Data
                                                                                                                                                                 14
18
                                                                                                                                                                 15
        . TDA124(vertex\_data\_in[124]), \\ . TDA126(vertex\_data\_in[126]), . TDA126(vertex\_data\_in[126]), . TDA126(vertex\_data\_in[126]), . TDA127(vertex\_data\_in[127]), // Write Test Data
20
                                                                                                                                                                 16
21
            //READ TEST SIGNALS
                                                                                                                                                                 17
22
             .BISTEB(vss),
                                                                                                                                                                 18
23
             .TOEB(vss).
                                                                                                                                                                 19
24
             .TMEB(vss),
                                                                                                                                                                20
       .TADRB0(pc_index[0]), .TADRB1
.TADRB3(pc_index[3]), // Read Test Address
                                                      .TADRB1(pc_index[1]),
                                                                                              .TADRB2(pc_index[2]),
                                                                                                                                                                21
26
                                                                                                                                                                22
27
28
        . TADRB4(pc\_index[4]), \ . TADRB5(pc\_index[5]), \ . TADRB6(pc\_index[6]), \ \ / / \ Read \ Test \ Address
29
             .AWTB(vss)
30
            );
31
        `endif // !`ifdef USE_BEHAVE_MEM
                                                           Page 13 of 14
                                                                                                                                                                                                                          Page 14 of 14
                                                                                        Ex. 2111 - param_cache_ctl.v
                                                                                                                                                                                                                                                        Ex. 2111 - param_cache_ctl.v
```

```
1 //SIdS
                                                                                                                      1 sclk global, srst, TP SP data0, TP SP data1, TP SP data2,
                                                                                                                           TP SP data3, TP SP data valid, TP SP gpr dst, TP SP gpr cmask,
 2 //$Change$
                                                                                                                            SQ\_SP\_instruct\_start, SQ\_SP\_instruct, SQ\_SP\_stall,
 4 // Filename : shader.v
                                                                                                                            SQ_SP_exp_pvalid, SQ_SP_exporting, SQ_SP_exp_id, SQ_SP_const,
 5 // Description : This module represents the Shader Pipe unit.
                                                                                                                           SO SP gpr wr addr. SO SP gpr rd addr. SO SP gpr rd en.
               : There are 4 instances of this module in the chip.
                                                                                                                           SQ_SP_gpr_wr_en, SQ_SP_gpr_phase_mux, SQ_SP_channel_mask,
               : Each shader pipe includes four vector units.
                                                                                                                           SQ_SP_pix_mask, SQ_SP_gpr_input_mux, SQ_SP_auto_count, SC_SP_data,
               : Each vector unit has four GPR/MAC instances and one scalar unit.
                                                                                                                            SC SP valid, SC SP type, SC SP last quad, SQ SP vsr data,
                                                                                                                             SQ\_SP\_vsr\_double, SQ\_SP\_vsr\_valid, SQ\_SP\_vsr\_read,
10 // Created On : Fri Nov 16 18:48:25 2001
                                                                                                                            SQ\_SP\_interp\_prim\_type, SQ\_SP\_interp\_ijline, SQ\_SP\_interp\_mode,
11 // Last Modified By:
                                                                                                                            SQ SP interp valid, SQ SP interp buff swap, SQ SP interp gen i0,
12 // Last Modified On:
                                                                                                                            CG_SP_pm_enb, SX_SP_vtx_data0, SX_SP_vtx_data1, SX_SP_vtx_data2
13 // Update Count : 0
                                                                                                                     13
                                                                                                                     14
14 // Status · Unknown Use with caution!
15
                                                                                                                     15
16 `timescale 1ns / 1ps
                                                                                                                     16
                                                                                                                                         sclk_global;
17
                                                                                                                     17
                                                                                                                            input srst;
18 module sp(/*AUTOARG*/
                                                                                                                     18
20 SP_TP_fetch_addr0, SP_TP_fetch_addr1, SP_TP_fetch_addr2,
                                                                                                                     20
                                                                                                                            assign sclk = sclk_global;
21
     SP TP fetch addr3, SP SX data0, SP SX data1, SP SX data2,
                                                                                                                     21
       SP_SX_data3, SP_SQ_const_addr, SP_SQ_valid, SP_SQ_kill_vect,
                                                                                                                            //SHADER(SP)-TEXTURE(TP)
     SP_SX_exp_pvalid, SP_SX_exporting, SP_SX_exp_alu_id,
                                                                                                                     23
                                                                                                                           //These buses represent the texture fetch request data.
24 SP SX exp dest,
                                                                                                                     24 //One 96 bit bus comming out of each vector unit....4 X 96 bits = 384 bits comming
25 // Inputs
                                                                                                                     25 //out of each shader pipe
                                           Page 1 of 19
                                                                                                                                                                Page 2 of 19
                                                                              Ex. 2112 - sp.v
                                                                                                                                                                                                   Ex. 2112 - sp.v
                                                                                                                      1 ati_dff_out #(128) usp_sx_data0(sclk, osp_sx_data0,q_sp_sx_data0);
                                                                         SP TP fetch addr0,
                                                                                                                      2 ati dff out #(128) usp_sx_data1(sclk, osp_sx_data1,q_sp_sx_data1);
     output [95:0]
SP_TP_fetch_addr1,SP_TP_fetch_addr2,SP_TP_fetch_addr3;
                                                                                                                            ati_dff_out #(128) usp_sx_data2(sclk, osp_sx_data2,q_sp_sx_data2);
       wire [95:0] sp_fetch_addr0, sp_fetch_addr1,sp_fetch_addr2,sp_fetch_addr3;
                                                                                                                            ati_dff_out #(128) usp_sx_data3(sclk, osp_sx_data3,q_sp_sx_data3);
       wire~[95:0]~q\_sp\_fetch\_addr0,~q\_sp\_fetch\_addr1, q\_sp\_fetch\_addr2, q\_sp\_fetch\_addr3;
                                                                                                                            //export data going out to SX (shader export)
      ati_dff_out #(96) usp_fetch_addr0(sclk,sp_fetch_addr0,q_sp_fetch_addr0);
                                                                                                                                      SP_SX_data0 = q_sp_sx_data0;
       ati\_dff\_out \# (96) \ usp\_fetch\_addr1 (sclk,sp\_fetch\_addr1,q\_sp\_fetch\_addr1); \\
                                                                                                                                         SP_SX_data1 = q_sp_sx_data1;
       ati_dff_out #(96) usp_fetch_addr2(sclk,sp_fetch_addr2,q_sp_fetch_addr2);
                                                                                                                                         SP SX data2 = q sp sx data2;
                                                                                                                            assign
       ati dff out #(96) usp fetch addr3(sclk,sp fetch addr3,q sp fetch addr3);
                                                                                                                                         SP\_SX\_data3 = q\_sp\_sx\_data3;
                                                                                                                     11
12
                                                                                                                     12
13
                  SP_TP_fetch_addr0 = q_sp_fetch_addr0;
      assign
14
                  SP\_TP\_fetch\_addr1 = q\_sp\_fetch\_addr1;
                                                                                                                     14
                                                                                                                           //The transfer on this interface is done via a MOVA instruction used to calculate the
15
       assign
                  SP_TP_fetch_addr2 = q_sp_fetch_addr2;
                                                                                                                     15
                                                                                                                            //address that SQ uses to access the Constant Store.
                  SP TP fetch addr3 = q sp fetch addr3;
16
       assign
                                                                                                                     17
                                                                                                                            output [35:0] SP SQ const addr;
18
                                                                                                                     18
                                                                                                                            output [0:0] SP SQ valid ;
      //SHADER(SP) - SX(SHADER EXPORT)
20 //This interface represents pixel/parameter data being exported out of the shader pipe
                                                                                                                     20
21 //into the SX block
                                                                                                                     21
22
                                                                                                                            //Kill return interface-----
       output [127:0] SP_SX_data0, SP_SX_data1,SP_SX_data2,SP_SX_data3;
                                                                                                                     23
       wire~[127:0]~~q\_sp\_sx\_data0,~q\_sp\_sx\_data1~,~q\_sp\_sx\_data2~,~q\_sp\_sx\_data3;\\
                                                                                                                     24
                                                                                                                            output [3:0] SP_SQ_kill_vect;
       wire [127:0] osp_sx_data0, osp_sx_data1, osp_sx_data2, osp_sx_data3;
                                                                                                                                                                 Page 4 of 19
                                           Page 3 of 19
                                                                              Ex. 2112 - sp.v
                                                                                                                                                                                                   Ex. 2112 - sp.v
```

ATI 2112 LG v. ATI IPR2015-00326

```
ati dff in #(1) tp data valid(sclk,TP SP data valid,q tp data valid);
 2
       //TEXTURE(TP)-SHADER(SP)
       //This interface represents the texture fetch return data
       //TP_SP_dst and TP_SP_cmask represent the GPR destination of this data
       //and a mask value per 32 channel respectivly.
       //TP_SP_data* is currently defined as a 128 bit bus.
                                                                                                                                    //SEQUENCER(SQ)-SHADER(SP)
       //This bus will propably be reduced to 64 bit.
                                                                                                                                    //the controls needed for writting
                                                                                                                                    //and reading the register files(GPRs).
        input [127:0] TP_SP_data0,TP_SP_data1,TP_SP_data2,TP_SP_data3;
                                                                                                                                    //ALU Instruction related data is also
        input [0:0] TP_SP_data_valid;
11
       input [6:0] TP_SP_gpr_dst;
12
       input [3:0] TP_SP_gpr_cmask;
                                                                                                                             12
                                                                                                                                    input [0:0] SQ_SP_instruct_start;
13
                                                                                                                             13
                                                                                                                                    input [20:0] SQ_SP_instruct;
14
       wire [127:0] q_tp_data0, q_tp_data1,q_tp_data2,q_tp_data3;
                                                                                                                                    input [0:0] SQ_SP_stall;
15
                                                                                                                                    input [3:0] SQ SP exp pvalid;
       wire [6:0] q tp gpr dst;
                                                                                                                             15
                                                                                                                                    input [0:0] SQ SP exporting;
16
       wire [3:0] q_tp_gpr_cmask;
                                                                                                                             16
17
                                                                                                                            17
                                                                                                                                    input [0:0] SQ SP exp id;
        wire [0:0] q_tp_data_valid;
18
                                                                                                                             18
19
       //registering the inputs from Texture pipe
                                                                                                                             19
                                                                                                                                    wire [0:0] q_sq_instruct_start;
20
       ati\_dff\_in\ \#(128)\ tp\_data0(sclk,TP\_SP\_data0,q\_tp\_data0);
                                                                                                                            20
                                                                                                                                    wire [20:0] q_sq_instruct;
21
       ati\_dff\_in\ \#(128)\ tp\_data1(sclk,TP\_SP\_data1,q\_tp\_data1);
                                                                                                                            21
                                                                                                                                    wire [0:0]
22
       ati\_dff\_in\ \#(128)\ tp\_data2(sclk,TP\_SP\_data2,q\_tp\_data2);
                                                                                                                                    wire [3:0]
23
       ati\_dff\_in~\#(128)~tp\_data3(sclk,TP\_SP\_data3,q\_tp\_data3);
                                                                                                                            23
                                                                                                                                                 q_sq_exporting;
24
       ati\_dff\_in~\#(7)~tp\_gpr\_dst(sclk,~TP\_SP\_gpr\_dst,~q\_tp\_gpr\_dst);\\
                                                                                                                            24
                                                                                                                                    wire [0:0] q_sq_exp_alu_id;
        ati\_dff\_in~\#(4)~tp\_gpr\_cmask(sclk, TP\_SP\_gpr\_cmask, q\_tp\_gpr\_cmask);
                                                                                                                            25
                                              Page 5 of 19
                                                                                                                                                                           Page 6 of 19
                                                                                   Ex. 2112 - sp.v
                                                                                                                                                                                                               Ex. 2112 - sp.v
                                                                                                                                    wire [0:0] q sp exp alu id;
                                                                                                                                    wire [5:0] q_sp_exp_dst;
       ati_dff_in #(1) sq_instruct_start(sclk,SQ_SP_instruct_start,q_sq_instruct_start);
       ati_dff_in #(21) sq_instruct(sclk,SQ_SP_instruct,q_sq_instruct);
                                                                                                                                    ati_dff_out #(4) usp_exp_pvalid(sclk,sp_exp_pvalid,q_sp_exp_pvalid);
       ati_dff_in #(1) sq_stall(sclk,SQ_SP_stall,q_sq_stall);
                                                                                                                                    ati\_dff\_out~\#(1)~usp\_exporting(sclk,sp\_exporting,q\_sp\_exporting);\\
       ati\_dff\_in\ \#(4)\ \ sq\_exp\_pvalid(sclk,SQ\_SP\_exp\_pvalid,q\_sq\_exp\_pvalid);
                                                                                                                                    ati\_dff\_out\ \#(1)\ usp\_exp\_alu\_id(sclk,sp\_exp\_alu\_id,q\_sp\_exp\_alu\_id);
        ati\_dff\_in \# (1) \ \ sq\_exporting(sclk, SQ\_SP\_exporting, q\_sq\_exporting);
                                                                                                                                    ati\_dff\_out\ \#(6)\ usp\_exp\_dst(sclk,sp\_exp\_dst,q\_sp\_exp\_dst);
        ati\_dff\_in \#(1) \ \ sq\_exp\_alu\_id(sclk,SQ\_SP\_exp\_id,q\_sq\_exp\_alu\_id);
                                                                                                                                    assign
                                                                                                                                                 SP_SX_exp_pvalid = q_sp_exp_pvalid;
10
                                                                                                                             10
                                                                                                                                    assign
                                                                                                                                                 SP_SX_exporting = q_sp_exporting;
11
       //The three buses below are pipelined through and outputed to SX
                                                                                                                            11
                                                                                                                                                  SP\_SX\_exp\_alu\_id = q\_sp\_exp\_alu\_id;
12
                                                                                                                            12
                                                                                                                                                  SP_SX_exp_dest = q_sp_exp_dst;
13
                                                                                                                            13
       output [3:0] SP_SX_exp_pvalid;
14
                                                                                                                                    //SEQUENCER(SQ)-SHADER(SP)
       output [0:0] SP_SX_exporting;
16
       output [0:0] SP_SX_exp_alu_id;
                                                                                                                                    //Constant Broadcast interface
17
       output [5:0] SP_SX_exp_dest;
18
                                                                                                                                    input [127:0] SQ_SP_const;
19
       wire [3:0] sp_exp_pvalid;
                                                                                                                                    wire [127:0] q_sq_const;
20
       wire [0:0] sp exporting;
                                                                                                                            20
21
                                                                                                                            21
       wire [0:0] sp exp alu id;
                                                                                                                                    ati dff in #(128) sq const(sclk,SQ SP const,q sq const);
22
                                                                                                                            22
       wire [5:0] sp_exp_dst;
23
                                                                                                                            23
24
       wire [3:0] q_sp_exp_pvalid;
                                                                                                                            24
       wire [0:0] q_sp_exporting;
                                                                                                                                    //SEQUENCER(SQ)-SHADER(SP)
                                              Page 7 of 19
                                                                                                                                                                           Page 8 of 19
                                                                                   Ex. 2112 - sp.v
                                                                                                                                                                                                               Ex. 2112 - sp.y
```



```
input [0:0] CG SP pm enb;
                                                                                                                                                                                                                                                                                                                 wire [127:0] Interpolated0, Interpolated1, Interpolated2, Interpolated3;
  2
                  wire [0:0] q_cg_sp_pm_enb;
                                                                                                                                                                                                                                                                                                                 wire [95:0] VertexIndex0, VertexIndex1, VertexIndex2, VertexIndex3;
                  ati_dff_in #(1) ucg_sp_pm_enb(sclk,CG_SP_pm_enb,q_cg_sp_pm_enb);
                 //SX - SP Interpolators : Parameter Cache return bus
                                                                                                                                                                                                                                                                                                                 //Interpolation Units--
                                                                                                                                                                                                                                                                                                                 interpolator\ uinterpolator(.oInterpolated0(Interpolated0),\ .oInterpolated1(Interpolated1),\ .oI
                  //data1 represents P1-P0
                  //data2 represents P2-P0
                                                                                                                                                                                                                                                                                                                                                                      .oInterpolated2(Interpolated2), .oInterpolated3(Interpolated3),
                  //remember : the difference engines are in the SX blocks
                                                                                                                                                                                                                                                                                                                                                                       .sx\_sp\_vtx\_data0(q\_sx\_vtx\_data0),
11
                                                                                                                                                                                                                                                                                                             .sx\_sp\_vtx\_delta10(q\_sx\_vtx\_data1),.sx\_sp\_vtx\_delta20(q\_sx\_vtx\_data2),
12
13
                  input [127:0] SX_SP_vtx_data0;
                                                                                                                                                                                                                                                                                                              . sq\_sp\_interp\_ijline (q\_sq\_interp\_ijline), . sq\_sp\_interp\_valid (q\_sq\_interp\_valid), . sq\_sp\_interp\_valid (q\_sq\_interp\_valid (q\_sq\_interp\_valid (q\_sq\_interp\_valid (q\_sq\_interp\_valid (q\_sq\_interp\_valid (q\_sq\_interp\_valid (q\_sq\_interp\_vali
14
                  input [131:0] SX_SP_vtx_data1, SX_SP_vtx_data2;
                                                                                                                                                                                                                                                                                               15
                                                                                                                                                                                                                                                                                                                                                                        .sq\_sp\_interp\_buff\_swap(q\_sq\_interp\_buff\_swap),
15
                                                                                                                                                                                                                                                                                                              .sc\_sp\_data(q\_sc\_data),.sc\_sp\_valid(q\_sc\_valid),.sq\_sp\_interp\_mode(q\_sq\_interp\_mode),\\
16
                  wire [127:0] q sx vtx data0;
                                                                                                                                                                                                                                                                                                18
                                                                                                                                                                                                                                                                                                                                                                      .sc\_sp\_type(q\_sc\_type),.sc\_sp\_quad\_last(q\_sc\_last\_quad),\\
17
                  wire \ [131:0] \quad q\_sx\_vtx\_data1, \ q\_sx\_vtx\_data2;
                                                                                                                                                                                                                                                                                                19
                                                                                                                                                                                                                                                                                                                                                      .sclk(sclk),.srst(srst));
18
                                                                                                                                                                                                                                                                                               20
19
                 //registering the inputs
                                                                                                                                                                                                                                                                                               21
20
                                                                                                                                                                                                                                                                                               22
21
                  ati dff in #(128) sx vtx data0(sclk,SX SP vtx data0,q sx vtx data0);
                                                                                                                                                                                                                                                                                               23
                                                                                                                                                                                                                                                                                                                 //Vertex Indices Staging registers and Control
22
                  ati_dff_in #(132) sx_vtx_data1(sclk,SX_SP_vtx_data1,q_sx_vtx_data1);
23
                  ati\_dff\_in\ \#(132)\ sx\_vtx\_data2(sclk,SX\_SP\_vtx\_data2,q\_sx\_vtx\_data2);
                                                                                                                                                                                                                                                                                               25
                                                                                                                                                                                                                                                                                                                sp vsr ctl usp vsr ctl(.ovtx index0(VertexIndex0), .ovtx index1(VertexIndex1),
24
                                                                                                                                                                                                                                                                                                                                                                .ovtx_index2(VertexIndex2), .ovtx_index3(VertexIndex3),
25
                                                                                                          Page 13 of 19
                                                                                                                                                                                                                                                                                                                                                                                                         Page 14 of 19
                                                                                                                                                                                               Ex. 2112 - sp.v
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              Ex. 2112 - sp.v
                                                                                                                                                                                                                                                                                                             . sq\_sp\_gpr\_phase\_mux(q\_sq\_gpr\_phase\_mux), \\ . sq\_sp\_channel\_mask(q\_sq\_channel\_mask), \\
                                                                  .isq vsr data(q sq vsr data), .isq vsr double(q sq vsr double),
                                                                  .isq vsr valid(q sq vsr valid), .isq vsr read(q sq vsr read),
                                                                                                                                                                                                                                                                                                               . sq\_sp\_pixel\_mask(q\_sq\_pix\_mask), \\ . sq\_sp\_gpr\_input\_mux(q\_sq\_gpr\_input\_mux), \\
                                                                  .sclk(sclk)..srst(srst)
                                                                                                                                                                                                                                                                                                                                                  . iInterpolated (Interpolated 0), \! /\! / \, iAuto Count,
                                                                                                                                                                                                                                                                                                                                                  .iVertexIndices(VertexIndex0),
                                                                                                                                                                                                                                                                                                                                                  .sq\_sp\_constant(q\_sq\_const),
                 //Instantiation of 4 Vector units (vector.v module)
                                                                                                                                                                                                                                                                                                                                                  .tp\_sp\_data(q\_tp\_data0),.tp\_sp\_gpr\_dst(q\_tp\_gpr\_dst),\\
                                                                                                                                                                                                                                                                                                                                                  .tp\_sp\_gpr\_cmask(q\_tp\_gpr\_cmask),.tp\_sp\_data\_valid(q\_tp\_data\_valid),\\
                                                                                                                                                                                                                                                                                                                                                  .sq\_sp\_exp\_pvalid(q\_sq\_exp\_pvalid),
10
                  vector uvector0(//outputs
                                                                                                                                                                                                                                                                                                                                                  .sq\_sp\_exporting(q\_sq\_exporting),
11
                                              .sp_sx_data(osp_sx_data0),
                                                                                                                                                                                                                                                                                                                                                  .sq_sp_exp_alu_id(q_sq_exp_alu_id)
12
                                                .sp_sx_exporting(sp_exporting),
13
                                                 .sp_sx_exp_dst(sp_exp_dst),
                                         .sp_sx_exp_alu_id(sp_exp_alu_id),
                                                                                                                                                                                                                                                                                               15
                                                  .sp_sx_exp_pvalid(sp_exp_pvalid),
                                                                                                                                                                                                                                                                                               16
                                                                                                                                                                                                                                                                                                                 vector uvector1(.sp sx data(osp sx data1),
                                                  .sp_tp_data(sp_fetch_addr0),
                                                                                                                                                                                                                                                                                               17
                                                                                                                                                                                                                                                                                                                                                  .sp_tp_data(sp_fetch_addr1),
17
                                                                                                                                                                                                                                                                                               18
                                                                                                                                                                                                                                                                                                                                                  .sq_sp_instruct_start(q_sq_instruct_start),
18
                                                                                                                                                                                                                                                                                               19
                                                                                                                                                                                                                                                                                                                                                  .sq\_sp\_instruct(q\_sq\_instruct),.sq\_sp\_stall(q\_sq\_stall),\\
                                                  .sq sp instruct start(q sq instruct start),
                                                                                                                                                                                                                                                                                               20
                                                                                                                                                                                                                                                                                                                                                  .sclk(sclk). .srst(srst).
20
                                                  .sq_sp_instruct(q_sq_instruct),.sq_sp_stall(q_sq_stall),
                                                                                                                                                                                                                                                                                               21
                                                                                                                                                                                                                                                                                                                                                  .sq\_sp\_wr\_addr(q\_sq\_gpr\_wr\_addr), .sq\_sp\_gpr\_rd\_addr(q\_sq\_gpr\_rd\_addr), \\
21
                                                  .sclk(sclk), .srst(srst),
                                                                                                                                                                                                                                                                                                             .sq\_sp\_mem\_rd\_ena(q\_sq\_gpr\_rd\_en),.sq\_sp\_mem\_wr\_ena(q\_sq\_gpr\_wr\_en),.sq\_sp\_wr\_ena(q\_sq\_gpr\_wr\_en),\\
22
                                                   .sq\_sp\_wr\_addr(q\_sq\_gpr\_wr\_addr), .sq\_sp\_gpr\_rd\_addr(q\_sq\_gpr\_rd\_addr), \\
                                                                                                                                                                                                                                                                                                             . sq\_sp\_gpr\_phase\_mux(q\_sq\_gpr\_phase\_mux), \\ . sq\_sp\_channel\_mask(q\_sq\_channel\_mask), \\
                                                                                                                                                                                                                                                                                                25
26
             . sq\_sp\_pixel\_mask(q\_sq\_pix\_mask), \\ . sq\_sp\_gpr\_input\_mux(q\_sq\_gpr\_input\_mux), \\
                                                                                                         Page 15 of 19
                                                                                                                                                                                                                                                                                                                                                                                                         Page 16 of 19
                                                                                                                                                                                               Ex. 2112 - sp.y
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               Ex. 2112 - sp.y
```

1	.iInterpolated(Interpolated1),// iAutoCount,	1 .sq_sp_instruct_start(q_sq_instruct_start),
2	.iVertexIndices(VertexIndex1),	2 .sq_sp_instruct(q_sq_instruct),.sq_sp_stall(q_sq_stall),
3	.sq_sp_constant(q_sq_const),	3 .sclk(sclk), .srst(srst),
4	.tp_sp_data(q_tp_data1),.tp_sp_gpr_dst(q_tp_gpr_dst),	4 .sq_sp_wr_addr(q_sq_gpr_wr_addr), .sq_sp_gpr_rd_addr(q_sq_gpr_rd_addr),
5	.tp_sp_gpr_cmask(q_tp_gpr_cmask),.tp_sp_data_valid(q_tp_data_valid));	5
6		6 .sq_sp_mem_rd_ena(q_sq_gpr_rd_en),.sq_sp_mem_wr_ena(q_sq_gpr_wr_en),.sq_sp_wr_ena(7 q_sq_gpr_wr_en),
7	vector uvector2(.sp_sx_data(osp_sx_data2),	8 .sq_sp_gpr_phase_mux(q_sq_gpr_phase_mux), 9 .sq_sp_channel_mask(a_sq_channel_mask).
8	.sp_tp_data(sp_fetch_addr2),	The state of the s
9	.sq_sp_instruct_start(q_sq_instruct_start),	10 .sq_sp_pixel_mask(q_sq_pix_mask), 11 .sq_sp_gpr_input_mux(q_sq_gpr_input_mux),
10	$.sq_sp_instruct(q_sq_instruct),.sq_sp_stall(q_sq_stall),$	12 .iInterpolated(Interpolated3),// iAutoCount,
11	.sclk(sclk), .srst(srst),	13 .iVertexIndices(VertexIndex3),
12	$.sq_sp_wr_addr(q_sq_gpr_wr_addr), .sq_sp_gpr_rd_addr(q_sq_gpr_rd_addr), \\$.sq_sp_constant(q_sq_const),
13 14	.sq_sp_mem_rd_ena(q_sq_gpr_rd_en),.sq_sp_mem_wr_ena(q_sq_gpr_wr_en),.sq_sp_wr_ena(.tp_sp_data(q_tp_data3),.tp_sp_gpr_dst(q_tp_gpr_dst),
15	q_sq_gpr_wr_en),	.tp_sp_gpr_cmask(q_tp_gpr_cmask),.tp_sp_data_valid(q_tp_data_valid));
16 17	.sq_sp_gpr_phase_mux(q_sq_gpr_phase_mux), .sq_sp_channel_mask(q_sq_channel_mask),	17
18	.sq_sp_pixel_mask(q_sq_pix_mask),	18 endmodule // sp
19	.sq_sp_gpr_input_mux(q_sq_gpr_input_mux),	19
20	.iInterpolated(Interpolated2),// iAutoCount,	20
21	.iVertexIndices(VertexIndex2),	21
22	.sq_sp_constant(q_sq_const),	22
23	$.tp_sp_data(q_tp_data2),.tp_sp_gpr_dst(q_tp_gpr_dst),\\$	23
24	$.tp_sp_gpr_cmask(q_tp_gpr_cmask),.tp_sp_data_valid(q_tp_data_valid));$	24
25		25
26	vector uvector3(.sp_sx_data(osp_sx_data3),	26
27	.sp_tp_data(sp_fetch_addr3),	27
	Page 17 of 19	Page 18 of 19
	Ex. 2112 - sp.v	Ex. 2112 - sp.v

Page 19 of 19 Ex. 2112 - sp.v

```
1 //
                      -*- Mode: Verilog -*-
 2 // Filename : export buffers.v
 3 \quad \  \  /\!\!\!/ \ Description \quad : This \ file \ represent \ the \ two \ banks \ of \ the \ export \ buffers \ (8 \ all \ together)
                                                                                                                           input [7:0] iread_addr;
              : and the data muxing at the output of the buffers into four distinct buses
                                                                                                                          input [6:0] iwrite_addr;
               one for each RB block
                                                                                                                     5 input [127:0] ipixel_data0.ipixel_data1.ipixel_data2.ipixel_data3:
 6 // Author : Andi Skende
                                                                                                                     6 input [127:0] ipixel_data4,ipixel_data5,ipixel_data6,ipixel_data7;
 7 // Created On : Tue Apr 16 17:13:53 2002
                                                                                                                                    sclk,srst;
 8 // Last Modified By:
                                                                                                                     8
                                                                                                                           input [0:0] imem wen, imem wew,imem re;
10 // Update Count : 0
                                                                                                                    10
                                                                                                                           //we need four phase counters for each rb request so we can serialize the data back to RBs
11 // Status : Open issues: generate the appropriate write and read enable controls for the 12 export buffers.
                                                                                                                    11
                                                                                                                           //into four consecutive cycles.
                                                                                                                    12
                                                                                                                          input [1:0] iphase_rb0;
                                                                                                                    13 input [1:0] iphase_rb1;
14 module export_buffers(/*AUTOARG*/
                                                                                                                    14 input [1:0] iphase rb2;
15 // Outputs
                                                                                                                    15
                                                                                                                           input [1:0] iphase_rb3;
      orb0_data, orb1_data, orb2_data, orb3_data, oclipp_data,
                                                                                                                    16
                                                                                                                           input [1:0] iphase_clipp;
       orb0_data_valid, orb1_data_valid, orb2_data_valid,
                                                                                                                    17
18 orb3_data_valid, oclipp_data_valid,
                                                                                                                    18
                                                                                                                          //valid read request from RBs ...stays high for four cycles.
                                                                                                                    19 //this signal is eventually pipelined out into orb#_data_valid
20 iread_addr, iwrite_addr, ipixel_data0, ipixel_data1, ipixel_data2,
                                                                                                                    20 input [0:0] iread_valid_rb0, iread_valid_rb1,iread_valid_rb2,iread_valid_rb3;
21 ipixel_data3, ipixel_data4, ipixel_data5, ipixel_data6,
                                                                                                                    21
                                                                                                                          input [0:0] iread valid clipp;
      ipixel data7, sclk, srst, imem wen, imem wew, imem re, iphase rb0,
                                                                                                                    23
24 iread valid rb1, iread valid rb2, iread valid rb3,
                                                                                                                           output [127:0] orb0 data, orb1 data, orb2 data, orb3 data;
25 iread valid clipp
                                                                                                                           output [127:0] oclipp_data;
                                          Page 1 of 84
                                                                                                                                                               Page 2 of 84
                                                                  Ex. 2113 - export buffers.v
                                                                                                                                                                                      Ex. 2113 - export buffers.v
 1 output [0:0] orb0_data_valid,orb1_data_valid,orb2_data_valid,orb3_data_valid;
                                                                                                                          assign sp_bank_sel3 = (iphase_rb3==0) ? iread_addr[7] : q_sp_bank_sel3;
      output [0:0] oclipp data valid;
                                                                                                                     2
      reg [0:0] rb0_data_valid,rb1_data_valid,rb2_data_valid,rb3_data_valid;
                                                                                                                     4
                                                                                                                           wire HIGH;
      reg [0:0] clipp data valid;
                                                                                                                           wire LOW;
                                                                                                                                     HIGH = 1'b1;
                                                                                                                          assign LOW = ~HIGH:
 8 //export buffers bank select
                                                                                                                     8
       wire sp bank sel;
               q_sp_bank_sel;
                                                                                                                           wire [127:0] buff0_out;
11
                                                                                                                    11
                                                                                                                           wire [127:0] buff1 out;
12
      wire sp_bank_sel0;
                                                                                                                          wire [127:0] buff2_out;
13 wire sp_bank_sel1;
                                                                                                                    13 wire [127:0] buff3_out;
14
      wire
                sp bank sel2;
                                                                                                                    14
                                                                                                                    15
15
                sp bank sel3;
                                                                                                                           wire [127:0] buff4 out;
      wire
16
                                                                                                                            wire [127:0] buff5_out;
17
                q_sp_bank_sel1;
                                                                                                                    17
                                                                                                                           wire [127:0] buff6 out:
                                                                                                                    18
                                                                                                                          wire [127:0] buff7 out;
18
      reg
               q sp bank sel2;
               q_sp_bank_sel3;
20
                                                                                                                    20 reg [127:0] rb0_data, rb1_data, rb2_data,rb3_data;
      assign sp bank sel = iread addr[7];
                                                                                                                    21
                                                                                                                          reg~[127:0] \quad q\_rb0\_data, q\_rb1\_data, q\_rb2\_data, q\_rb3\_data;
21
23
      assign sp_bank_sel0 = (iphase_rb0==0) ? iread_addr[7] : q_sp_bank_sel0;
                                                                                                                           reg [127:0] bank0_data0;
                sp bank sel1 = (iphase rb1==0)? iread addr[7]: q sp bank sel1;
                                                                                                                          reg [127:0] bank0 data1:
      assign sp_bank_sel2 = (iphase_rb2==0) ? iread_addr[7] : q_sp_bank_sel2;
                                                                                                                          reg [127:0] bank0_data2;
                                           Page 3 of 84
                                                                                                                                                               Page 4 of 84
                                                                 Ex. 2113 - export buffers.v
                                                                                                                                                                                      Ex. 2113 - export buffers.v
```

ATI 2113 LG v. ATI IPR2015-00326

```
reg [127:0] bank0 data3;
                                                                                                                                          q phase rb0 <= iphase rb0;
        reg [127:0] bank0_clipp_data;
                                                                                                                                          q\_phase\_rb1 \mathrel{<=} iphase\_rb1;
 2
                                                                                                                                          q\_phase\_rb2 <= iphase\_rb2;
        reg [127:0] bank1_data0;
        reg [127:0] bank1_data1;
                                                                                                                                          q_phase_rb3 <= iphase_rb3;
        reg [127:0] bank1_data2;
                                                                                                                                          q_phase_clipp <= iphase_clipp;
        reg [127:0] bank1_data3;
                                                                                                                                          q0_read_valid_rb0 <= iread_valid_rb0;
        reg [127:0] bank1_clipp_data;
                                                                                                                                          q1\_read\_valid\_rb0 \mathrel{<=} q0\_read\_valid\_rb0;
                                                                                                                                          q2\_read\_valid\_rb0 <= q1\_read\_valid\_rb0;
                                                                                                                                          q3_read_valid_rb0 <= q2_read_valid_rb0;
       reg~[1:0]~q\_phase\_rb0~,~q\_phase\_rb1~,~q\_phase\_rb2~,~q\_phase\_rb3;\\
                                                                                                                                          q0\_read\_valid\_rb1 <= iread\_valid\_rb1;
11
       reg [1:0] q_phase_clipp;
                                                                                                                                          q1\_read\_valid\_rb1 \mathrel{<=} q0\_read\_valid\_rb1;
12
                                                                                                                                          q2\_read\_valid\_rb1 <= q1\_read\_valid\_rb1;
13
       reg [0:0] q0_read_valid_rb0, q1_read_valid_rb0;
                                                                                                                                          q3\_read\_valid\_rb1 <= q2\_read\_valid\_rb1;
14
       reg [0:0] q2_read_valid_rb0, q3_read_valid_rb0;
                                                                                                                                          q0_read_valid_rb2 <= iread_valid_rb2;
15
                                                                                                                                          ql_read_valid_rb2 <= q0_read_valid_rb2;
       reg [0:0] q0 read valid rb1, q1 read valid rb1;
                                                                                                                             15
                                                                                                                                          q2_read_valid_rb2 <= q1_read_valid_rb2;
16
       reg [0:0] q2 read valid rb1, q3 read valid rb1;
                                                                                                                             16
17
                                                                                                                             17
                                                                                                                                          q3 read valid rb2 <= q2 read valid rb2:
       reg [0:0] q0 read valid rb2, q1 read valid rb2;
18
       reg [0:0] q2 read valid rb2, q3 read valid rb2;
                                                                                                                             18
                                                                                                                                          q0 read valid rb3 <= iread valid rb3:
19
       reg [0:0] q0_read_valid_rb3, q1_read_valid_rb3;
                                                                                                                             19
                                                                                                                                          q1 read valid rb3 <= q0 read valid rb3;
                                                                                                                                          q2 read valid rb3 <= q1 read valid rb3;
20
       reg [0:0] q2_read_valid_rb3, q3_read_valid_rb3;
                                                                                                                             20
                                                                                                                                          q3\_read\_valid\_rb3 \mathrel{<=} q2\_read\_valid\_rb3;
21
       reg~[0:0]~q0\_read\_valid\_clipp,~q1\_read\_valid\_clipp;
                                                                                                                             21
22
       reg~[0:0]~q2\_read\_valid\_clipp,~q3\_read\_valid\_clipp;
                                                                                                                             22
                                                                                                                                          q0 read valid clipp <= iread valid clipp;
23
                                                                                                                             23
                                                                                                                                          q1\_read\_valid\_clipp \mathrel{<=} q0\_read\_valid\_clipp;
24
        always @(posedge sclk)
                                                                                                                             24
                                                                                                                                          q2\_read\_valid\_clipp \mathrel{<=} q1\_read\_valid\_clipp;
25
        begin
                                                                                                                                          q3_read_valid_clipp <= q2_read_valid_clipp;
                                              Page 5 of 84
                                                                                                                                                                           Page 6 of 84
                                                                       Ex. 2113 - export_buffers.v
                                                                                                                                                                                                    Ex. 2113 - export_buffers.v
        end
                                                                                                                                     always @(posedge sclk)
 2
                                                                                                                              2
                                                                                                                                     begin
                                                                                                                                          q write addr <= iwrite addr.
 4
       reg [7:0]
                     q0 read addr, q1 read addr,q2 read addr;
                                                                                                                                          q mem wew <= imem wew;
       reg [0:0]
                     q0_mem_re, q1_mem_re, q2_mem_re;
                                                                                                                                          q_mem_wen <= imem_wen;
                                                                                                                                      end
                                                                                                                              7
       //skewing the read enable and read address buses
        always @(posedge sclk)
                                                                                                                                     wire [6:0] read_mem_addr;
10
                                                                                                                             10
                                                                                                                                     assign read_mem_addr = iread_addr[6:0];
11
             q0_read_addr <= iread_addr;
                                                                                                                             11
12
             q1_read_addr <= q0_read_addr;
                                                                                                                             12
                                                                                                                                 'ifdef USE BEHAVE MEM
13
             q2\_read\_addr \le q1\_read\_addr;
                                                                                                                                     dum_mem_p2 #(7,128) bank0_buff0(.iRCLK(sclk),
14
             q0_mem_re <= imem_re;
                                                                                                                             14
                                                                                                                                                                .iWCLK(sclk),
15
                                                                                                                             15
                                                                                                                                                                  .iMER(imem_re),
            q1_mem_re <= q0_mem_re;
16
                                                                                                                                                                  .iMEW(imem_wen),
             q2\_mem\_re \mathrel{<=} q1\_mem\_re;
17
                                                                                                                                                                 .iWEN(imem_wew),
18
                                                                                                                                                                  .iRADR(iread_addr[6:0]),
19
                                                                                                                                                                  .iWADR(iwrite_addr),
20
                                                                                                                             20
                                                                                                                                                                  .iD(ipixel data0),
       //skewing the write enable and write address buses so they line up with
                                                                                                                                                                  .oQ(buff0_out)
21
                                                                                                                             21
       //the data bus
22
                                                                                                                             22
23
                                                                                                                             23
                                                                                                                                   'else // !'ifdef USE BEHAVE MEM
       reg [6:0] q write addr;
24
                                                                                                                             24
       reg~[0:0]~q\_mem\_wew,~q\_mem\_wen;
25
                                                                                                                                     rfsd2 80x128cm2sw0 ubank0 buff0
                                              Page 7 of 84
                                                                                                                                                                           Page 8 of 84
                                                                                                                                                                                                    Ex. 2113 - export buffers.v
                                                                       Ex. 2113 - export buffers.v
```

1 2	(/*VRGIO rfsd2_80x128cm2sw0 ipixel_data0 buff0_out iwrite_addr iread_addr imem_wen imem re null*/	1 .QB44(buff0_out[44]), .QB45(buff0_out[45]), .QB46(buff0_out[46]), 2 .QB47(buff0_out[47]), // Read Data
3	// READ INTERFACE	3 .QB48(buff0_out[48]), .QB49(buff0_out[49]), .QB50(buff0_out[50]), 4 .QB51(buff0_out[51]), .// Read Data
4 5	.CLKB(iSCLK), // Read Clock	5 .QB52(buff0_out[52]), .QB53(buff0_out[53]), .QB54(buff0_out[54]),
6	.OEB(vdd), // Output enable .MEB(imem_re), // Read enable	6 .QB55(buff0_out[55]), // Read Data 7 .QB56(buff0_out[56]), .QB57(buff0_out[57]), .QB58(buff0_out[58]),
7 8	.ADRB0(iread_addr[0]), .ADRB1(iread_addr[1]), .ADRB2(iread_addr[2]), .ADRB3(iread_addr[3]), // Read Address	8 .QB59(buff0_out[59]), // Read Data 9 .QB60(buff0_out[60]), .QB61(buff0_out[61]), .QB62(buff0_out[62]),
9 10	.ADRB4(iread_addr[4]), .ADRB5(iread_addr[5]), .ADRB6(iread_addr[6]), // Read Address	10 .QB63(buff0_out[63]), // Read Data
11	.QB0(buff0_out[0]), .QB1(buff0_out[1]), .QB2(buff0_out[2]), .QB3(buff0_out[3]), // Read	11
12 13	Data .QB4(buff0_out[4]), .QB5(buff0_out[5]), .QB6(buff0_out[6]), .QB7(buff0_out[7]), // Read	13 .QB68(buff0_out[68]), .QB69(buff0_out[69]), .QB70(buff0_out[70]), .QB71(buff0_out[71]), // Read Data
14 15	Data .QB8(buff0_out[8]), .QB9(buff0_out[9]), .QB10(buff0_out[10]), .QB11(buff0_out[11]), //	15 .QB72(buff0_out[72]), .QB73(buff0_out[73]), .QB74(buff0_out[74]), .QB75(buff0_out[75]), // Read Data
16	Read Data	17 .QB76(buff0_out[76]), .QB77(buff0_out[77]), .QB78(buff0_out[78]), 18 .QB79(buff0_out[79]), // Read Data
17 18	QB12(buff0_out[12]), QB13(buff0_out[13]), QB14(buff0_out[14]), QB15(buff0_out[15]), // Read Data	19 .QB80(buff0_out[80]), .QB81(buff0_out[81]), .QB82(buff0_out[82]), 20 .QB83(buff0_out[83]), // Read Data
19 20	.QB16(buff0_out[16]), .QB17(buff0_out[17]), .QB18(buff0_out[18]), .QB19(buff0_out[19]), // Read Data	21 .QB84(buff0_out[84]), .QB85(buff0_out[85]), .QB86(buff0_out[86]), 22 .QB87(buff0_out[87]), // Read Data
21 22	$. QB20(buff0_out[20]), \qquad . QB21(buff0_out[21]), \qquad . QB22(buff0_out[22]), \\ . QB23(buff0_out[23]), \ // Read \ Data$	23 .QB88(buff0_out[88]), .QB89(buff0_out[89]), .QB90(buff0_out[90]),
23 24	.QB24(buff0_out[24]), .QB25(buff0_out[25]), .QB26(buff0_out[26]), .QB27(buff0_out[27]), // Read Data	24 .QB91(buff0_out[91]), // Read Data 25 .QB92(buff0_out[92]), .QB93(buff0_out[93]), .QB94(buff0_out[94]),
25 26	.QB28(buff0_out[28]), .QB29(buff0_out[29]), .QB30(buff0_out[30]), .QB31(buff0_out[31]), // Read Data	26 .QB95(buff0_out[95]), // Read Data 27 .QB96(buff0_out[96]), .QB97(buff0_out[97]), .QB98(buff0_out[98]),
27	.QB32(buff0_out[32]), .QB33(buff0_out[33]), .QB34(buff0_out[34]),	28 .QB99(buff0_out[99]), // Read Data
28 29	.QB35(buff0_out[35]), // Read Data .QB36(buff0_out[36]), .QB37(buff0_out[37]), .QB38(buff0_out[38]),	29 QB100(buff0_out[100]), QB101(buff0_out[101]), QB102(buff0_out[102]), 30 QB103(buff0_out[103]), // Read Data
30 31	.QB39(buff0_out[39]), // Read Data .QB40(buff0_out[40]), .QB41(buff0_out[41]), .QB42(buff0_out[42]),	31 .QB104(buff0_out[104]), .QB105(buff0_out[105]), .QB106(buff0_out[106]), 32 .QB107(buff0_out[107]), // Read Data
32	.QB43(buff0_out[43]), // Read Data	Page 10 of \$4
	Page 9 of 84 Ex. 2113 - export_buffers.v	Page 10 of 84 Ex. 2113 - export_buffers.v
1 2 3 4 4 5 6 6 7 7 8 8 9 10 111 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26	QB108(buff0_out[108), QB109(buff0_out[109), QB110(buff0_out[110), QB111(buff0_out[111)), // Read Data QB112(buff0_out[115)), // Read Data QB12(buff0_out[115)), // Read Data QB12(buff0_out[115)), // Read Data QB120(buff0_out[116)), QB119(buff0_out[117)), // Read Data QB120(buff0_out[120)), // Read Data QB120(buff0_out[120)), // Read Data QB120(buff0_out[120)), // Read Data QB123(buff0_out[123)), // Read Data QB124(buff0_out[124]), QB125(buff0_out[125]), // QB123(buff0_out[124]), // Read Data QB124(buff0_out[124]), // Read Data QB124(buff0_out[127]), // Read Data // WRITE INTERFACE CLKA(iSCLK), // Write Clock WEA(imem_wen), // Write enable MEA(vdd), // Memory enable ADRA0(iwrite_addr[0]), ADRA1(iwrite_addr[1]), ADRA5(iwrite_addr[4]), ADRA5(iwrite_addr[4]), ADRA5(iwrite_addr[4]), ADRA6(iwrite_addr[6]), // Write Address DA0(ipixel_data0[1]), DA1(ipixel_data0[1]), DA2(ipixel_data0[2]), DA3(ipixel_data0[1]), // Write Data DA4(ipixel_data0[1]), // Write Data DA8(ipixel_data0[1]), // Write Data DA9(ipixel_data0[1]), // Write Data DA9(ipixel_dat	DA28(ipixel_data0[31]), // Write Data DA32(ipixel_data0[32]), // Write Data DA32(ipixel_data0[32]), // Write Data DA32(ipixel_data0[32]), // Write Data DA35(ipixel_data0[32]), // Write Data DA36(ipixel_data0[35]), // Write Data DA36(ipixel_data0[36]), // Write Data DA36(ipixel_data0[38]), // Write Data DA36(ipixel_data0[39]), // Write Data DA40(ipixel_data0[43]), // Write Data DA40(ipixel_data0[43]), // Write Data DA44(ipixel_data0[43]), // Write Data DA44(ipixel_data0[43]), // Write Data DA44(ipixel_data0[43]), // Write Data DA48(ipixel_data0[43]), // Write Data DA48(ipixel_data0[43]), // Write Data DA48(ipixel_data0[43]), // Write Data DA54(ipixel_data0[43]), // Write Data DA55(ipixel_data0[51]), // Write Data DA55(ipixel_data0[52]), // Write Data DA55(ipixel_data0[53]), // Write Data DA56(ipixel_data0[53]), // Write Data DA60(ipixel_data0[53]), // Write Data DA60(ipixel_data0[53]), // Write Data DA64(ipixel_data0[63]), // Write Data DA64(ipixel_data0[63]), // Write Data DA64(ipixel_data0[63]), // Write Data DA64(ipixel_data0[63]), // Write Data DA66(ipixel_data0[63]), // Write Data DA66(ipixel_data0[63]), // Write Data DA66(ipixel_data0[63]), // Write Data DA66(ipixel_data0[63]), // Write Data DA72(ipixel_data0[63]), // Write Data DA72(ipixel_data0[71]), // Write Data DA72(ipixel_data0[72]), // Write Data DA72(ipixel_data0[73]), // Write Data
27 28	.DA16(pixel_data0[16]), .DA17(pixel_data0[17]), .DA18(pixel_data0[18]), .DA19(pixel_data0[19]), // Write Data	29 .DA84(ipixel_data0[84]), .DA85(ipixel_data0[85]), .DA86(ipixel_data0[86]), 30 .DA87(ipixel_data0[87]), // Write Data
29 30	$.DA20(ipixel_data0[20]), \qquad .DA21(ipixel_data0[21]), \qquad .DA22(ipixel_data0[22]), \\ .DA23(ipixel_data0[23]), \ // Write Data$	31 .DA88(ipixel_data0[88]), .DA89(ipixel_data0[89]), .DA90(ipixel_data0[90]),
31 32	$.DA24(ipixel_data0[24]), .DA25(ipixel_data0[25]), .DA26(ipixel_data0[26]), \\ .DA27(ipixel_data0[27]), \ // Write Data$	32 .DA91(ipixel_data0[91]), // Write Data
	Page 11 of 84 Ex. 2113 - export_buffers.v	Page 12 of 84 Ex. 2113 - export_buffers.v

1 2	.DA92(ipixel_data0[92]), .DA93(ipixel_data0[93]), .DA94(ipixel_data0[94]), .DA95(ipixel_data0[95]), // Write Data	1 .TDA12(ipixel_data0[12]), .TDA13(ipixel_data0[13]), .TDA14(ipixel_data0[14]), 2 .TDA15(ipixel_data0[15]), // Write Test Data
3 4	.DA98(ipixel_data0[98]), .DA97(ipixel_data0[97]), .DA98(ipixel_data0[98]), .DA99(ipixel_data0[98]), .DA99(ipixel_data0[98	3TDA16(ipixel_data0[16]),TDA17(ipixel_data0[17]),TDA18(ipixel_data0[18]), 4TDA19(ipixel_data0[19]), // Write Test Data
5	.DA100(ipixel_data0[100]), .DA101(ipixel_data0[101]), .DA102(ipixel_data0[102]), .DA103(ipixel_data0[103]), // Write Data	5TDA20(ipixel_data0[20]),TDA21(ipixel_data0[21]),TDA22(ipixel_data0[22]), 6TDA23(ipixel_data0[23]), // Write Test Data
7	.DA104(ipixel_data0[104]), .DA105(ipixel_data0[105]), .DA106(ipixel_data0[106]),	7 .TDA24(ipixel_data0[24]), .TDA25(ipixel_data0[25]), .TDA26(ipixel_data0[26]),
9	.DA107(ipixel_data0[107]), // Write Data .DA108(ipixel_data0[108]),DA109(ipixel_data0[109]),DA110(ipixel_data0[110]),	8 .TDA27(ipixel_data0[27]), // Write Test Data 9 .TDA28(ipixel_data0[28]), .TDA29(ipixel_data0[29]), .TDA30(ipixel_data0[30]),
10 11	.DA111(ipixel_data0[111]), // Write Data .DA112(ipixel_data0[112]),DA113(ipixel_data0[113]),DA114(ipixel_data0[114]),	10 .TDA31(ipixel_data0[31]), // Write Test Data 11 .TDA32(ipixel_data0[32]), .TDA33(ipixel_data0[33]), .TDA34(ipixel_data0[34]),
12 13	.DA115(ipixel_data0[115]), // Write Data .DA116(ipixel_data0[116]),DA117(ipixel_data0[117]),DA118(ipixel_data0[118]),	12 .TDA35(ipixel_data0[35]), // Write Test Data 13 .TDA36(ipixel_data0[36]), .TDA37(ipixel_data0[37]), .TDA38(ipixel_data0[38]),
14 15	.DA119(ipixel_data0[119]), //Write Data .DA120(ipixel_data0[120]),DA121(ipixel_data0[121]),DA122(ipixel_data0[122]),	14 .TDA39(ipixel_data0[39]), // Write Test Data 15 .TDA40(ipixel_data0[40]), .TDA41(ipixel_data0[41]), .TDA42(ipixel_data0[42]),
16	.DA123(ipixel_data0[123]), // Write Data	16 .TDA43(ipixel_data0[43]), // Write Test Data
17 18	.DA124(ipixel_data0[124]), .DA125(ipixel_data0[125]), .DA126(ipixel_data0[126]), .DA127(ipixel_data0[127]), // Write Data	17 .TDA44(ipixel_data0[44]), .TDA45(ipixel_data0[45]), .TDA46(ipixel_data0[46]), .TDA47(ipixel_data0[47]), // Write Test Data
19 20	// WRITE TEST SIGNALS .BISTEA(vss),	19 .TDA48(ipixel_data0[48]), .TDA49(ipixel_data0[49]), .TDA50(ipixel_data0[50]), 20 .TDA51(ipixel_data0[51]), // Write Test Data
21	.TWEA(vss), // Test write enable	21 .TDA52(ipixel_data0[52]), .TDA53(ipixel_data0[53]), .TDA54(ipixel_data0[54]), 22 .TDA55(ipixel_data0[55]), // Write Test Data
22 23	.TMEA(vss), // Test memory enable .TADRA0(iwrite_addr[0]), .TADRA1(iwrite_addr[1]), .TADRA2(iwrite_addr[2]),	23 .TDA56(ipixel_data0[56]), .TDA57(ipixel_data0[57]), .TDA58(ipixel_data0[58]), 24 .TDA59(ipixel_data0[59]), // Write Test Data
24 25	.TADRA4(iwrite addr[4]), ./ Write Test Address .TADRA4(iwrite addr[4]), .TADRA5(iwrite addr[5]), .TADRA6(iwrite addr[6]), ./ Write	25TDA60(ipixel_data0[60]),TDA61(ipixel_data0[61]),TDA62(ipixel_data0[62]), 26TDA63(ipixel_data0[63]), // Write Test Data
26 27	Test Address	27TDA64(ipixel_data0[64]),TDA65(ipixel_data0[65]),TDA66(ipixel_data0[66]), 28TDA67(ipixel_data0[67]), // Write Test Data
28	.TDA3(ipixel_data0[3]), // Write Test Data	29 .TDA68(ipixel_data0[68]), .TDA69(ipixel_data0[69]), .TDA70(ipixel_data0[70]), 30 .TDA71(ipixel_data0[71]), // Write Test Data
29 30	.TDA4(ipixel_data0[4]), .TDA5(ipixel_data0[5]), .TDA6(ipixel_data0[6]), .TDA7(ipixel_data0[7]), // Write Test Data	31TDA72(ipixel_data0[72]),TDA73(ipixel_data0[73]),TDA74(ipixel_data0[74]), 32TDA75(ipixel_data0[75]), // Write Test Data
31 32	.TDA8(ipixel_data0[8]), .TDA9(ipixel_data0[9]), .TDA10(ipixel_data0[10]), .TDA11(ipixel_data0[11]), // Write Test Data	
	Page 13 of 84 Ex. 2113 - export_buffers.v	Page 14 of 84 Ex. 2113 - export_buffers.v
1 2 3 4 4 5 5 6 6 7 7 8 8 9 9 10 11 12 12 13 13 14 15 16 16 17 18 19 20 21 22 23 24 24 25 26 27 28 29	.TDA76(ipixel_data0[76]), .TDA77(ipixel_data0[77]), .TDA78(ipixel_data0[78]), .TDA79(ipixel_data0[78]), .TDA80(ipixel_data0[80]), .TDA81(ipixel_data0[81]), .TDA83(ipixel_data0[83]), .// Write Test Data .TDA83(ipixel_data0[83]), .// Write Test Data .TDA83(ipixel_data0[84]), .TDA85(ipixel_data0[85]), .TDA87(ipixel_data0[87]), .// Write Test Data .TDA83(ipixel_data0[87]), .// Write Test Data .TDA83(ipixel_data0[87]), .// Write Test Data .TDA93(ipixel_data0[97]), .// Write Test Data .TDA94(ipixel_data0[97]), .// Write Test Data .TDA94(ipixel_data0[97]), .// Write Test Data .TDA96(ipixel_data0[96]), .TDA97(ipixel_data0[97]), .TDA99(ipixel_data0[99]), .// Write Test Data .TDA96(ipixel_data0[99]), .// Write Test Data .TDA100(ipixel_data0[103]), .// Write Test Data .TDA104(ipixel_data0[103]), .// Write Test Data .TDA104(ipixel_data0[103]), .// Write Test Data .TDA104(ipixel_data0[104]), .TDA105(ipixel_data0[105]), .TDA107(ipixel_data0[107]), .// Write Test Data .TDA108(ipixel_data0[108]), .TDA109(ipixel_data0[109]), .TDA110(ipixel_data0[106]), .TDA111(ipixel_data0[111]), .// Write Test Data .TDA112(ipixel_data0[111]), .// Write Test Data .TDA112(ipixel_data0[112]), .// Write Test Data .TDA112(ipixel_data0[112]), .// Write Test Data .TDA112(ipixel_data0[112]), .// Write Test Data .TDA124(ipixel_data0[123]), .// Wr	1
30 31	.TMEB(vss), .TADRB0(iread_addr[0]), .TADRB1(iread_addr[1]), .TADRB2(iread_addr[2]),	24 'else // l'ifdef USE_BEHAVE_MEM 25 rfsd2_80x128cm2sw0 ubank0_buff1
32	.TADRB3(iread_addr[3]), // Read Test Address Page 15 of 84 Ex. 2113 - export_buffers.v	Page 16 of 84 Ex. 2113 - export_buffers.v

1 2	(/*VRGIO rfsd2_80x128cm2sw0 ipixel_data1 buff0_out iwrite_addr q0_read_addr imem_wen q0_mem_re null*/	1 .QB44(buff0_out[44]), .QB45(buff0_out[45]), .QB46(buff0_out[46]), 2 .QB47(buff0_out[47]), // Read Data
3	// READ INTERFACE	3 .QB48(buff0_out[48]), .QB49(buff0_out[49]), .QB50(buff0_out[50]), 4 .QB51(buff0_out[51]), // Read Data
4	.CLKB(iSCLK), // Read Clock .OEB(vdd), // Output enable	5 .QB52(buff0_out[52]), .QB53(buff0_out[53]), .QB54(buff0_out[54]),
6	.MEB(q0_mem_re), // Read enable	6 .QB55(buff0_out[55]), // Read Data 7 .QB56(buff0_out[56]), .QB57(buff0_out[57]), .QB58(buff0_out[58]),
7 8	$. ADRB0(q0_read_addr[0]), .ADRB1(q0_read_addr[1]), .ADRB2(q0_read_addr[2]), \\ .ADRB3(q0_read_addr[3]), // Read Address$	8 .QB59(buff0_out[59]), // Read Data 9 .QB60(buff0_out[60]), .QB61(buff0_out[61]), .QB62(buff0_out[62]),
9 10	.ADRB4(q0_read_addr[4]), .ADRB5(q0_read_addr[5]), .ADRB6(q0_read_addr[6]), // Read Address	10 .QB63(buff0_out[63]), // Read Data 11 .QB64(buff0_out[64]), .QB65(buff0_out[65]), .QB66(buff0_out[66]),
11 12	.QB0(buff0_out[0]), .QB1(buff0_out[1]), .QB2(buff0_out[2]), .QB3(buff0_out[3]), $//$ Read Data	12 .QB67(buff0_out[67]), // Read Data
13	.QB4(buff0_out[4]), .QB5(buff0_out[5]), .QB6(buff0_out[6]), .QB7(buff0_out[7]), // Read	13 QB68(buff0_out[68]), QB69(buff0_out[69]), QB70(buff0_out[70]), 14 QB71(buff0_out[71]), // Read Data
14 15	Data .QB8(buff0_out[8]), .QB9(buff0_out[9]), .QB10(buff0_out[10]), .QB11(buff0_out[11]), //	15 .QB72(buff0_out[72]), .QB73(buff0_out[73]), .QB74(buff0_out[74]), .QB75(buff0_out[75]), // Read Data
16 17	Read Data .QB12(buff0_out[12]),	17 .QB76(buff0_out[76]), .QB77(buff0_out[77]), .QB78(buff0_out[78]), 18 .QB79(buff0_out[79]), // Read Data
18 19	.QB15(buff0_out[15]),	19 .QB80(buff0_out[80]), .QB81(buff0_out[81]), .QB82(buff0_out[82]), 20 .QB83(buff0_out[83]), // Read Data
20	.QB19(buff0_out[19]), // Read Data	21 .QB84(buff0_out[84]), .QB85(buff0_out[85]), .QB86(buff0_out[86]), 22 .QB87(buff0_out[87]), // Read Data
21 22	.QB20(buff0_out[20]), .QB21(buff0_out[21]), .QB22(buff0_out[22]), .QB23(buff0_out[23]), // Read Data	23 .QB88(buff0_out[88]), .QB89(buff0_out[89]), .QB90(buff0_out[90]), 24 .QB91(buff0_out[91]), // Read Data
23 24	.QB24(buff0_out[24]), .QB25(buff0_out[25]), .QB26(buff0_out[26]), .QB27(buff0_out[27]), // Read Data	25 .QB92(buff0_out[92]), .QB93(buff0_out[93]), .QB94(buff0_out[94]), 26 .QB95(buff0_out[95]), // Read Data
25 26	$.QB28(buff0_out[28]), \qquad .QB29(buff0_out[29]), \qquad .QB30(buff0_out[30]), \\ .QB31(buff0_out[31]), \ // Read \ Data$	27 .QB96(buff0_out[96]), .QB97(buff0_out[97]), .QB98(buff0_out[98]),
27 28	$.QB32(buff0_out[32]), \qquad .QB33(buff0_out[33]), \qquad .QB34(buff0_out[34]), \\ .QB35(buff0_out[35]), \ // Read \ Data$	28 .QB99(buff0_out[99]), // Read Data 29 .QB100(buff0_out[100]), .QB101(buff0_out[101]), .QB102(buff0_out[102]),
29 30	.QB36(buff0_out[36]), .QB37(buff0_out[37]), .QB38(buff0_out[38]), .QB39(buff0_out[39]), // Read Data	30 .QB103(buff0_out[103]), // Read Data 31 .QB104(buff0_out[104]), .QB105(buff0_out[105]), .QB106(buff0_out[106]),
31 32	.QB40(buff0_out[40]), .QB41(buff0_out[41]), .QB42(buff0_out[42]), .QB43(buff0_out[43]), // Read Data	32 .QB107(buff0_out[107]), // Read Data
	Page 17 of 84 Ex. 2113 - export_buffers.v	Page 18 of 84 Ex. 2113 - export_buffers.v
1 2 3	.QB108(buff0_out[108]), .QB109(buff0_out[109]), .QB110(buff0_out[110]), .QB111(buff0_out[111]), // Read Data .QB112(buff0_out[112]), .QB13(buff0_out[113]), .QB114(buff0_out[114]),	1 .DA28(ipixel_data1[28]), .DA29(ipixel_data1[29]), .DA30(ipixel_data1[30]), 2 .DA31(ipixel_data1[31]), // Write Data 3 .DA32(ipixel_data1[32]), .DA33(ipixel_data1[33]), .DA34(ipixel_data1[34]),
4	.QB115(buff0_out[115]), // Read Data .QB116(buff0_out[116]),QB117(buff0_out[117]),QB118(buff0_out[118]),	4 .DA35(ipixel_data1[35]), // Write Data 5 .DA36(ipixel_data1[36]), .DA37(ipixel_data1[37]), .DA38(ipixel_data1[38]),
6	.QB120(buff0_out[120]),	6 .DA39(ipixel_data1[39]), // Write Data 7 .DA40(ipixel_data1[40]), .DA41(ipixel_data1[41]), .DA42(ipixel_data1[42]),
8	.QB123(buff0_out[123]), // Read Data	8 .DA43(ipixel_data1[43]), // Write Data
9 10	.QB124(buff0_out[124]), .QB125(buff0_out[125]), .QB126(buff0_out[126]), .QB127(buff0_out[127]), // Read Data	9 .DA44(ipixel_data1[44]), .DA45(ipixel_data1[45]), .DA46(ipixel_data1[46]), 10 .DA47(ipixel_data1[47]), // Write Data
11	// WRITE INTERFACE .CLKA(iSCLK), // Write Clock	11 .DA48(ipixel_data1[48]), .DA49(ipixel_data1[49]), .DA50(ipixel_data1[50]),
13	.WEA(imem_wen), // Write enable	13 .DA52(ipixel_data1[52]), .DA53(ipixel_data1[53]), .DA54(ipixel_data1[54]), .DA55(ipixel_data1[55]), // Write Data
14	.MEA(vdd), // Memory enable	15 .DA56(ipixel_data1[56]), .DA57(ipixel_data1[57]), .DA58(ipixel_data1[58]), 16 .DA59(ipixel_data1[59]), // Write Data
15 16	.ADRA0(iwrite_addr[0]), .ADRA1(iwrite_addr[1]), .ADRA2(iwrite_addr[2]), .ADRA3(iwrite_addr[3]), // Write Address	17 .DA60(ipixel_data1[60]), .DA61(ipixel_data1[61]), .DA62(ipixel_data1[62]), 18 .DA63(ipixel_data1[63]), // Write Data
17 18	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6(iwrite_addr[6]), // Write Address	19 .DA64(ipixel_data1[64]), .DA65(ipixel_data1[65]), .DA66(ipixel_data1[66]), 20 .DA67(ipixel_data1[67]), // Write Data
19 20	.DA0(ipixel_data1[0]), .DA1(ipixel_data1[1]), .DA2(ipixel_data1[2]), .DA3(ipixel_data1[3]), // Write Data	21 .DA68(ipixel_data1[68]), .DA69(ipixel_data1[69]), .DA70(ipixel_data1[70]),
21 22	$.DA4(ipixel_data1[4]), \qquad .DA5(ipixel_data1[5]), \qquad .DA6(ipixel_data1[6]), \\ .DA7(ipixel_data1[7]), \ // Write \ Data$	22 .DA71(ipixel_data1[71]), // Write Data 23 .DA72(ipixel_data1[72]), .DA73(ipixel_data1[73]), .DA74(ipixel_data1[74]),
23 24	$.DA8(ipixel_data1[8]), \qquad .DA9(ipixel_data1[9]), \qquad .DA10(ipixel_data1[10]), \\ .DA11(ipixel_data1[11]), \ // Write Data$	24 .DA75(ipixel_data1[75]), // Write Data 25 .DA76(ipixel_data1[76]), .DA77(ipixel_data1[77]), .DA78(ipixel_data1[78]),
25 26	.DA12(ipixel_data1[12]), .DA13(ipixel_data1[13]), .DA14(ipixel_data1[14]), .DA15(ipixel_data1[15]), // Write Data	26 .DA79(ipixel_data1[79]), // Write Data 27 .DA80(ipixel_data1[80]), .DA81(ipixel_data1[81]), .DA82(ipixel_data1[82]),
27 28	.DA16(pixel_data1[18]), .DA17(ipixel_data1[17]), .DA18(ipixel_data1[18]), .DA19(ipixel_data1[19]), // Write Data	28 .DA83(ipixel_data1[83]), // Write Data
29	.DA20(ipixel_data1[20]),	30 .DA87(ipixel_data1[87]), // Write Data
30		31 .DA88(ipixel_data1[88]), .DA89(ipixel_data1[89]), .DA90(ipixel_data1[90]), 32 .DA91(ipixel_data1[91]), // Write Data
31	.DA24(ipixel data1[24]), .DA25(ipixel data1[25]), .DA26(ipixel data1[26]),	32 .DAM(place_datal(M)), // white batta
31 32		Page 20 of 84 Ex. 2113 - export_buffers.v

1 2	.DA92(ipixel_data1[92]), .DA93(ipixel_data1[93]), .DA95(ipixel_data1[95]), // Write Data	.DA94(ipixel_data1[94]),	1 2	.TDA12(ipixel_data1[12]), .TDA13(ipixel_data1[13] .TDA15(ipixel_data1[15]), // Write Test Data), .TDA14(ipixel_data1[14]),
3 4	.DA96(ipixel_data1[96]), .DA97(ipixel_data1[97]), .DA99(ipixel_data1[99]), // Write Data	.DA98(ipixel_data1[98]),	3 4	.TDA16(ipixel_data1[16]), .TDA17(ipixel_data1[17].TDA19(ipixel_data1[19]), // Write Test Data), .TDA18(ipixel_data1[18]),
5	.DA100(ipixel_data1[100]), .DA101(ipixel_data1[101]),	.DA102(ipixel_data1[102]),	5	.TDA20(ipixel_data1[20]), .TDA21(ipixel_data1[21]), .TDA22(ipixel_data1[22]),
7	.DA103(ipixel_data1[103]), // Write Data .DA104(ipixel_data1[104]), .DA105(ipixel_data1[105]),	.DA106(ipixel_data1[106]),	7	.TDA23(ipixel_data1[23]), // Write Test Data .TDA24(ipixel_data1[24]), .TDA25(ipixel_data1[25])), .TDA26(ipixel_data1[26]),
8	.DA107(ipixel_data1[107]), // Write Data .DA108(ipixel_data1[108]), .DA109(ipixel_data1[109]),	.DA110(ipixel data1[110]),	8	.TDA27(ipixel_data1[27]), // Write Test Data .TDA28(ipixel_data1[28]), .TDA29(ipixel_data1[29])), .TDA30(ipixel_data1[30]),
10 11	.DA111(ipixel_data1[111]), // Write Data .DA112(ipixel_data1[112]), .DA113(ipixel_data1[113]),	.DA114(ipixel data1[114]),	10	.TDA31(ipixel_data1[31]), // Write Test Data .TDA32(ipixel_data1[32]), .TDA33(ipixel_data1[33])	
12	.DA115(ipixel_data1[115]), // Write Data		12	.TDA35(ipixel_data1[35]), // Write Test Data	
13 14	.DA116(ipixel_data1[116]), .DA117(ipixel_data1[117]), .DA119(ipixel_data1[119]), // Write Data	.DA118(ipixel_data1[118]),	13 14	.TDA36(ipixel_data1[36]), .TDA37(ipixel_data1[37] .TDA39(ipixel_data1[39]), // Write Test Data), .TDA38(ipixel_data1[38]),
15 16	.DA120(ipixel_data1[120]), .DA121(ipixel_data1[121]), .DA123(ipixel_data1[123]), // Write Data	.DA122(ipixel_data1[122]),	15 16	.TDA40(ipixel_data1[40]), .TDA41(ipixel_data1[41] .TDA43(ipixel_data1[43]), // Write Test Data), .TDA42(ipixel_data1[42]),
17 18	.DA124(ipixel_data1[124]),DA125(ipixel_data1[125]), .DA127(ipixel_data1[127]), // Write Data	.DA126(ipixel_data1[126]),	17 18	.TDA44(ipixel_data1[44]), .TDA45(ipixel_data1[45] .TDA47(ipixel_data1[47]), // Write Test Data), .TDA46(ipixel_data1[46]),
19	// WRITE TEST SIGNALS		19 20	.TDA48(ipixel_data1[48]), .TDA49(ipixel_data1[49] .TDA51(ipixel_data1[51]), // Write Test Data), .TDA50(ipixel_data1[50]),
20 21	.BISTEA(vss), .TWEA(vss), // Test write enable		21 22	.TDA52(ipixel_data1[52]), .TDA53(ipixel_data1[53] .TDA55(ipixel_data1[55]), // Write Test Data), .TDA54(ipixel_data1[54]),
22	.TMEA(vss), // Test memory enable		23	.TDA56(ipixel datal[56]), .TDA57(ipixel datal[57]), .TDA58(ipixel_data1[58]),
23 24	.TADRA0(iwrite_addr[0]), .TADRA1(iwrite_addr[1]), .TADRA3(iwrite_addr[3]), // Write Test Address	.TADRA2(iwrite_addr[2]),	24 25	.TDA59(ipixel_data1[59]), // Write Test Data .TDA60(ipixel_data1[60]), .TDA61(ipixel_data1[61]), .TDA62(ipixel_data1[62]),
25 26	.TADRA4(iwrite_addr[4]), .TADRA5(iwrite_addr[5]), .TADI Test Address	RA6(iwrite_addr[6]), // Write	26 27	.TDA63(ipixel_data1[63]), // Write Test Data .TDA64(ipixel_data1[64]), .TDA65(ipixel_data1[65])), .TDA66(ipixel_data1[66]),
27 28	.TDA0(ipixel_data1[0]), .TDA1(ipixel_data1[1]), .TDA3(ipixel_data1[3]), // Write Test Data	.TDA2(ipixel_data1[2]),	28	.TDA67(ipixel_data1[67]), // Write Test Data	
29	.TDA4(ipixel data1[4]), .TDA5(ipixel data1[5]),	.TDA6(ipixel_data1[6]),	30	.TDA71(ipixel_data1[71]), // Write Test Data	
30	.TDA7(ipixel_data1[7]), // Write Test Data .TDA8(ipixel_data1[8]), .TDA9(ipixel_data1[9]),	.TDA10(ipixel data1[10]),	31 32	.TDA72(ipixel_data1[72]), .TDA73(ipixel_data1[73] .TDA75(ipixel_data1[75]), // Write Test Data), .TDA74(ipixel_data1[74]),
32	.TDA11(ipixel_data1[11]), // Write Test Data	· · · · · · · · · · · · · · · · · · ·			
34			I		
32	Page 21 of 84	Ex. 2113 - export_buffers.v		Page 22 of 84	Ex. 2113 - export_buffers.v
1 2 3 4 5 6	Page 21 of 84 .TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[79]), // Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), // Write Test Data .TDA84(ipixel_data1[84]),	Ex. 2113 - export_buffers.v .TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]),	1 2 3 4 5	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif'/!'ifdef USE_BEHAVE_MEM	
1 2 3 4 5	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[79]), // Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), // Write Test Data .TDA84(ipixel_data1[84]), .TDA85(ipixel_data1[85]),	.TDA78(ipixel_datal[78]), .TDA82(ipixel_datal[82]),	3 4 5 6	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss));	
1 2 3 4 5 6	.TDA76(ipixel_datal[76]), .TDA77(ipixel_datal[77]), .TDA79(ipixel_datal[77]), .Write Test Data .TDA80(ipixel_datal[80]), .TDA81(ipixel_datal[81]), .TDA83(ipixel_datal[83]), .Write Test Data .TDA84(ipixel_datal[84]), .TDA85(ipixel_datal[85]), .TDA87(ipixel_datal[87]), .Write Test Data .TDA88(ipixel_datal[88]), .TDA89(ipixel_datal[89]),	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]),	3 4 5	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss));	
1 2 3 4 4 5 6 6 7 8 9 10 11 11	.TDA76(ipixel_data1[76]),TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[79]), // Write Test Data .TDA80(ipixel_data1[80]),TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[81]), // Write Test Data .TDA84(ipixel_data1[84]),TDA85(ipixel_data1[85]), .TDA87(ipixel_data1[87]), // Write Test Data .TDA88(ipixel_data1[88]),TDA99(ipixel_data1[89]), .TDA91(ipixel_data1[91]), // Write Test Data .TDA92(ipixel_data1[92]),TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), .TDA95(ipixel_data1[95]),TDA95(ipixel_data1[95]),TDA95(ipixel_data1[96]),TDA97(ipixel_data1[97]),	.TDA78(ipixel_datal[78]), .TDA82(ipixel_datal[82]), .TDA86(ipixel_datal[86]), .TDA90(ipixel_datal[90]),	2 3 3 4 5 5 6 7 7 8 8 9	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM	
1 2 3 4 4 5 6 6 7 8 8 9 10 11 12 13	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[87]), ./Write Test Data .TDA80(ipixel_data1[83]), ./Write Test Data .TDA84(ipixel_data1[84]), ./Write Test Data .TDA84(ipixel_data1[87]), ./Write Test Data .TDA88(ipixel_data1[88]),TDA89(ipixel_data1[89]), .TDA91(ipixel_data1[91]), ./Write Test Data .TDA92(ipixel_data1[92]),TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[93]), .TDA95(ipixel_data1[94]), .TDA97(ipixel_data1[97]), .TDA96(ipixel_data1[96]), .TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[96]), .TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[96]), .TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[96]), .TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[96]), .TDA97(ipixel_data1[96]), .TDA97(ipixel_data1[96]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]), .TDA101(ipixel_data1[101]),	.TDA78(ipixel_datal[78]), .TDA82(ipixel_datal[82]), .TDA86(ipixel_datal[86]), .TDA90(ipixel_datal[90]), .TDA94(ipixel_datal[94]), .TDA98(ipixel_datal[98]),	2 3 4 5 6 7 8	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); `endif // ! ifdef USE_BEHAVE_MEM	
1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 13 14 15 15	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[79]), // Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), // Write Test Data .TDA84(ipixel_data1[83]), // Write Test Data .TDA84(ipixel_data1[87]), // Write Test Data .TDA85(ipixel_data1[88]), .TDA85(ipixel_data1[89]), .TDA97(ipixel_data1[91]), // Write Test Data .TDA92(ipixel_data1[91]), // Write Test Data .TDA92(ipixel_data1[92]), .TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), // Write Test Data .TDA96(ipixel_data1[96]), .TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[193]), // Write Test Data .TDA100(ipixel_data1[103]), // Write Test Data .TDA100(ipixel_data1[103]), // Write Test Data .TDA104(ipixel_data1[103]), // Write Test Data .TDA104(ipixel_data1[103]), // Write Test Data	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]),	2 3 4 5 6 7 8 9	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM	
1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 16 17 7	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[87]), Write Test Data .TDA80(ipixel_data1[83]), TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), TDA85(ipixel_data1[85]), TDA85(ipixel_data1[85]), TDA85(ipixel_data1[85]), TDA85(ipixel_data1[85]), TDA93(ipixel_data1[89]), TDA93(ipixel_data1[91]), Write Test Data .TDA92(ipixel_data1[92]), TDA93(ipixel_data1[93]), TDA93(ipixel_data1[93]), TDA93(ipixel_data1[93]), TDA93(ipixel_data1[93]), TDA94(ipixel_data1[103]), TDA94(ipixel_data1[103]), TDA101(ipixel_data1[103]), TDA104(ipixel_data1[103]), TDA105(ipixel_data1[103]), TDA105(ipixel_data1[103]), TDA105(ipixel_data1[105]), TDA105(ipixel_d	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]),	2 3 4 5 6 7 8 9 10 11 12	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk),	
1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 16 17 18	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[87]), .Write Test Data .TDA80(ipixel_data1[83]),TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]),TDA83(ipixel_data1[84]),TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[87]),	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]),	2 3 4 5 6 7 8 9 10	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'iifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'iifdef USE_BEHAVE_MEM	
1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), ./Write Test Data .TDA84(ipixel_data1[83]), ./Write Test Data .TDA87(ipixel_data1[87]), ./Write Test Data .TDA87(ipixel_data1[87]), ./Write Test Data .TDA96(ipixel_data1[91]), ./Write Test Data .TDA92(ipixel_data1[92]), .TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[93]), .TDA95(ipixel_data1[93]), .TDA95(ipixel_data1[94]), .TDA96(ipixel_data1[95]), .Write Test Data .TDA96(ipixel_data1[100]), .TDA101(ipixel_data1[101]), .TDA103(ipixel_data1[101]), .TDA103(ipixel_data1[107]), .TDA104(ipixel_data1[107]), .TDA105(ipixel_data1[105]), .TDA106(ipixel_data1[107]), .TDA106(ipixel_data1[10	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]), .TDA114(ipixel_data1[114]),	2 3 4 5 6 7 8 9 10 11 12 13	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk), .iWCLK(sclk),	
1 2 2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 22	TDA76(ipixel_data1[76]),TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[79]), // Write Test Data TDA80(ipixel_data1[83]), // Write Test Data TDA81(ipixel_data1[83]), // Write Test Data TDA84(ipixel_data1[84]),TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA95(ipixel_data1[87]), // Write Test Data .TDA92(ipixel_data1[92]), // Write Test Data .TDA92(ipixel_data1[92]), // Write Test Data .TDA95(ipixel_data1[96]), // Write Test Data .TDA96(ipixel_data1[96]), .TDA97(ipixel_data1[97]), .TDA95(ipixel_data1[99]), // Write Test Data .TDA100(ipixel_data1[103]), // Write Test Data .TDA104(ipixel_data1[103]), // Write Test Data .TDA104(ipixel_data1[104]), .TDA105(ipixel_data1[105]), .TDA107(ipixel_data1[107]), // Write Test Data .TDA108(ipixel_data1[111]), // Write Test Data .TDA112(ipixel_data1[111]), // Write Test Data .TDA112(ipixel_data1[111]), // Write Test Data .TDA116(ipixel_data1[111]), // Write Test Data	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[102]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[110]), .TDA114(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(iRCLK(sclk), .iWCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen), .iWEN(imem_wew),	
1 2 2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[83]), .Write Test Data .TDA84(ipixel_data1[84]),TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[87]),Write Test Data .TDA83(ipixel_data1[88]),TDA93(ipixel_data1[89]), .TDA91(ipixel_data1[91]),Write Test Data .TDA92(ipixel_data1[91]),TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]),TDA95(ipixel_data1[95]),TDA95(ipixel_data1[95]),TDA95(ipixel_data1[105]), .TDA95(ipixel_data1[105]), .TDA104(ipixel_data1[105]), .TDA103(ipixel_data1[105]), .TDA104(ipixel_data1[107]),Write Test Data .TDA104(ipixel_data1[107]),Write Test Data .TDA104(ipixel_data1[108]), .TDA109(ipixel_data1[109]), .TDA111(ipixel_data1[11]),TDA113(ipixel_data1[113]), .TDA113(ipixel_data1[113]), .TDA115(ipixel_data1[115]),TDA117(ipixel_data1[117]), .TDA117(ipixel_data1[117]),TDA117(ipix	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]), .TDA1122(ipixel_data1[122]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(iRCLK(sclk), .iWCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen),	
1 2 2 3 3 4 4 5 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 23	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[83]), .Write Test Data .TDA84(ipixel_data1[84]), TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[85]), .TDA85(ipixel_data1[87]), Write Test Data .TDA98(ipixel_data1[88]), TDA98(ipixel_data1[89]), .TDA91(ipixel_data1[91]), TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), TDA93(ipixel_data1[97]), .TDA95(ipixel_data1[95]), TDA97(ipixel_data1[97]), .TDA95(ipixel_data1[103]), .TDA101(ipixel_data1[101]), .TDA103(ipixel_data1[103]), TDA104(ipixel_data1[103]), TDA104(ipixel_data1[107]), Write Test Data TDA104(ipixel_data1[104]), TDA105(ipixel_data1[105]), .TDA107(ipixel_data1[107]), Write Test Data TDA112(ipixel_data1[111]), TDA113(ipixel_data1[113]), .TDA113(ipixel_data1[113]), .TDA115(ipixel_data1[113]), TDA117(ipixel_data1[113]), .TDA117(ipixel_data1[113]), TDA117(ipixel_data1[117]),	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]), .TDA1122(ipixel_data1[122]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif//!'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk), .iMCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen), .iWEN(imem_wew), .iRADR(q1_read_addr[6:0]), .iWADR(iwrite_addr), .iD(ipixel_data2),	
1 1 2 2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 22 24 25 26 27	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), Write Test Data .TDA84(ipixel_data1[83]), Write Test Data .TDA87(ipixel_data1[87]), Write Test Data .TDA87(ipixel_data1[98]), TDA89(ipixel_data1[89]), .TDA91(ipixel_data1[99]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), TDA95(ipixel_data1[97]), .TDA96(ipixel_data1[96]), TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[101]), TDA96(ipixel_data1[101]), .TDA104(ipixel_data1[103]), TDA104(ipixel_data1[103]), TDA105(ipixel_data1[105]), .TDA107(ipixel_data1[107]), TDA109(ipixel_data1[107]), TDA119(ipixel_data1[113]), TDA119(ipixel_data1[113]), TDA119(ipixel_data1[115]), TDA113(ipixel_data1[113]), TDA116(ipixel_data1[116]), TDA117(ipixel_data1[117]), TDA119(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[112]), TDA112(ipixel_data1[121]), TDA112(ipixel_data1[121]), TDA112(ipixel_data1[121]), TDA112(ipixel_data1[122]), TDA112(ipixel_data1[123]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA1125(ipixel_data1[125]), TDA1	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]), .TDA1122(ipixel_data1[122]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif//!'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk), .iWCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen), .iWEN(imem_wew), .iRADR(q1_read_addr[6:0]), .iWADR(iwrite_addr),	
1 1 2 2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), ./Write Test Data .TDA84(ipixel_data1[83]), ./Write Test Data .TDA87(ipixel_data1[87]), ./Write Test Data .TDA87(ipixel_data1[88]), .TDA89(ipixel_data1[89]), .TDA97(ipixel_data1[97]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[93]), .TDA95(ipixel_data1[93]), .TDA95(ipixel_data1[93]), .TDA96(ipixel_data1[93]), .Write Test Data .TDA96(ipixel_data1[103]), .TDA101(ipixel_data1[101]), .TDA103(ipixel_data1[103]), .TDA101(ipixel_data1[103]), .TDA104(ipixel_data1[103]), .TDA104(ipixel_data1[103]), .TDA105(ipixel_data1[103]), .TDA105(ipixel_data1[113]), .TDA105(ipixel_data1[113]), .TDA105(ipixel_data1[113]), .TDA115(ipixel_data1[113]), .TDA115(ipixel_data1[113]), .TDA115(ipixel_data1[113]), .TDA115(ipixel_data1[113]), .TDA116(ipixel_data1[113]), .TDA116(ipixel_data1[113]), .TDA117(ipixel_data1[113]), .TDA116(ipixel_data1[113]), .TDA117(ipixel_data1[113]), .TDA112(ipixel_data1[121]), .TDA112(ipixel_data1[121]), .TDA112(ipixel_data1[123]), .TDA112	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]), .TDA1122(ipixel_data1[122]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif//!'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk), .iWCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen), .iWEN(imem_wen), .iWEN(imem_wen), .iWADR(iwrite_addr), .iD(ipixel_data2), .oQ(buff2_out)	
1 1 2 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), Write Test Data .TDA84(ipixel_data1[83]), Write Test Data .TDA87(ipixel_data1[87]), Write Test Data .TDA87(ipixel_data1[98]), TDA89(ipixel_data1[89]), .TDA91(ipixel_data1[99]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), TDA95(ipixel_data1[97]), .TDA96(ipixel_data1[96]), TDA97(ipixel_data1[97]), .TDA99(ipixel_data1[101]), TDA96(ipixel_data1[101]), .TDA104(ipixel_data1[103]), TDA104(ipixel_data1[103]), TDA105(ipixel_data1[105]), .TDA107(ipixel_data1[107]), TDA109(ipixel_data1[107]), TDA119(ipixel_data1[113]), TDA119(ipixel_data1[113]), TDA119(ipixel_data1[115]), TDA113(ipixel_data1[113]), TDA116(ipixel_data1[116]), TDA117(ipixel_data1[117]), TDA119(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[119]), TDA112(ipixel_data1[112]), TDA112(ipixel_data1[121]), TDA112(ipixel_data1[121]), TDA112(ipixel_data1[121]), TDA112(ipixel_data1[122]), TDA112(ipixel_data1[123]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA112(ipixel_data1[125]), TDA1125(ipixel_data1[125]), TDA1	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA110(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]), .TDA1122(ipixel_data1[122]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk), .iWCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen), .iWEN(imem_wen), .iWADR(iwrite_addr), .iD(ipixel_data2), .oQ(buff2_out)); 'else // !'ifdef USE_BEHAVE_MEM rfsd2_80x128cm2sw0 ubank0_buff2	, .TADRB6(q0_read_addr[6]), //
1 2 3 3 4 4 5 6 6 7 8 8 9 10 11 12 13 13 14 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28 29	.TDA76(ipixel_data1[76]), .TDA77(ipixel_data1[77]), .TDA79(ipixel_data1[77]), .Write Test Data .TDA80(ipixel_data1[80]), .TDA81(ipixel_data1[81]), .TDA83(ipixel_data1[83]), ./Write Test Data .TDA84(ipixel_data1[83]), ./Write Test Data .TDA87(ipixel_data1[87]), ./Write Test Data .TDA88(ipixel_data1[88]), .TDA89(ipixel_data1[89]), .TDA97(ipixel_data1[97]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA93(ipixel_data1[93]), .TDA95(ipixel_data1[95]), ./Write Test Data .TDA96(ipixel_data1[95]), ./Write Test Data .TDA100(ipixel_data1[100]), .TDA101(ipixel_data1[101]), .TDA90(ipixel_data1[103]), ./Write Test Data .TDA104(ipixel_data1[104]), .TDA105(ipixel_data1[105]), .TDA107(ipixel_data1[107]), ./Write Test Data .TDA108(ipixel_data1[108]), .TDA109(ipixel_data1[109]), .TDA111(ipixel_data1[117]), .TDA1115(ipixel_data1[117]), ./Write Test Data .TDA112(ipixel_data1[112]), ./Write Test Data .TDA116(ipixel_data1[113]), ./Write Test Data .TDA116(ipixel_data1[113]), ./Write Test Data .TDA119(ipixel_data1[113]), ./Write Test Data .TDA119(ipixel_data1[113]), ./Write Test Data .TDA119(ipixel_data1[113]), ./Write Test Data .TDA120(ipixel_data1[113]), ./Write Test Data .TDA120(ipixe	.TDA78(ipixel_data1[78]), .TDA82(ipixel_data1[82]), .TDA86(ipixel_data1[86]), .TDA90(ipixel_data1[90]), .TDA94(ipixel_data1[94]), .TDA98(ipixel_data1[98]), .TDA102(ipixel_data1[102]), .TDA106(ipixel_data1[106]), .TDA114(ipixel_data1[110]), .TDA114(ipixel_data1[114]), .TDA118(ipixel_data1[118]), .TDA122(ipixel_data1[122]), .TDA126(ipixel_data1[122]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.TADRB4(q0_read_addr[4]), .TADRB5(q0_read_addr[5] Read Test Address .AWTB(vss)); 'endif//!'ifdef USE_BEHAVE_MEM wire [6:0] q1_read_mem_addr; assign q1_read_mem_addr = q1_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff2(.iRCLK(sclk), .iMCLK(sclk), .iMER(q1_mem_re), .iMEW(imem_wen), .iWEN(imem_wen), .iWEN(imem_wen), .iWADR(iwrite_addr], .iD(ipixel_data2), .oQ(buff2_out)); 'else // l'ifdef USE_BEHAVE_MEM	, .TADRB6(q0_read_addr[6]), //

1	// READ INTERFACE	1 .QB48(buff2_out[48]), .QB49(buff2_out[49]), .QB50(buff2_out[50]), 2 .QB51(buff2_out[51]), // Read Data
2	.CLKB(iSCLK), // Read Clock .OEB(vdd), // Output enable	3 .QB52(buff2_out[52]), .QB53(buff2_out[53]), .QB54(buff2_out[54]), 4 .QB55(buff2_out[55]), // Read Data
4	.MEB(q1_mem_re), // Read enable	5 .QB56(buff2_out[56]), .QB57(buff2_out[57]), .QB58(buff2_out[58]), 6 .QB59(buff2_out[59]), // Read Data
5 6	.ADRB0(q1_read_addr[0]), .ADRB1(q1_read_addr[1]), .ADRB2(q1_read_addr[2]), .ADRB3(q1_read_addr[3]), // Read Address	7
7 8	$. ADRB4(q1_read_addr[4]), . ADRB5(q1_read_addr[5]), . ADRB6(q1_read_addr[6]), // \\ Read \; Address$	9 .QB64(buff2_out[64]), .QB65(buff2_out[65]), .QB66(buff2_out[66]),
9 10	.QB0(buff2_out[0]), .QB1(buff2_out[1]), .QB2(buff2_out[2]), .QB3(buff2_out[3]), // Read Data	10 .QB67(buft2_out[67]), // Read Data 11 .QB68(buft2_out[68]), .QB69(buft2_out[69]), .QB70(buft2_out[70]),
11 12	$. QB4(buff2_out[4]), \ . QB5(buff2_out[5]), \ . QB6(buff2_out[6]), \ . QB7(buff2_out[7]), \ //\ Read\ Data$	12 .QB71(buff2_out[71]), // Read Data 13 .QB72(buff2_out[72]), .QB73(buff2_out[73]), .QB74(buff2_out[74]),
13 14	.QB8(buff2_out[8]), .QB9(buff2_out[9]), .QB10(buff2_out[10]), .QB11(buff2_out[11]), $\ //$ Read Data	14 .QB75(buff2_out[75]), // Read Data 15 .QB76(buff2_out[76]), .QB77(buff2_out[77]), .QB78(buff2_out[78]),
15 16	.QB12(buff2_out[12]), .QB13(buff2_out[13]), .QB14(buff2_out[14]), .QB15(buff2_out[15]), // Read Data	16 .QB79(buff2_out[79]), // Read Data 17 .QB80(buff2_out[80]), .QB81(buff2_out[81]), .QB82(buff2_out[82]),
17 18	.QB16(buff2_out[16]), .QB17(buff2_out[17]), .QB18(buff2_out[18]), .QB19(buff2_out[19]), // Read Data	18 .QB83(buff2_out[83]), // Read Data 19 .QB84(buff2_out[84]), .QB85(buff2_out[85]), .QB86(buff2_out[86]),
19 20	.QB20(buff2_out[20]), .QB21(buff2_out[21]), .QB22(buff2_out[22]), .QB23(buff2_out[23]), // Read Data	20 .QB87(buff2_out[87]), // Read Data 21 .QB88(buff2_out[88]), .QB89(buff2_out[89]), .QB90(buff2_out[90]),
21 22	.QB24(buff2_out[24]), .QB25(buff2_out[25]), .QB26(buff2_out[26]), .QB27(buff2_out[27]), // Read Data	22 .QB91(buff2_out[91]), // Read Data 23 .QB92(buff2_out[92]), .QB93(buff2_out[93]), .QB94(buff2_out[94]),
23 24	.QB28(buff2_out[28]), .QB29(buff2_out[29]), .QB30(buff2_out[30]), .QB31(buff2_out[31]), // Read Data	24 .QB95(buff2_out[95]), // Read Data 25 .QB96(buff2_out[96]), .QB97(buff2_out[97]), .QB98(buff2_out[98]),
25 26	$. QB32(buff2_out[32]), \\ . QB33(buff2_out[33]), \\ . QB34(buff2_out[34]), \\ . QB35(buff2_out[35]), \\ // Read\ Data$	26 .QB99(buff2_out[99]), // Read Data 27 .QB100(buff2_out[100]), .QB101(buff2_out[101]), .QB102(buff2_out[102]), 28 .OB103(buff2_out[103]), // Read Data
27 28	.QB36(buff2_out[36]), .QB37(buff2_out[37]), .QB38(buff2_out[38]), .QB39(buff2_out[39]), // Read Data	26Q8104(buff2_out[104]),QB105(buff2_out[105]),QB106(buff2_out[106]),QB107(buff2_out[107]),QB105(buff2_out[105]),QB106(buff2_out[106]),QB107(buff2_out[107]),QB106(buff2_out[106]),QB106(buff2_
29 30	.QB40(buft2_out[40]), .QB41(buft2_out[41]), .QB42(buft2_out[42]), .QB43(buft2_out[43]), // Read Data	31 .QB108(buff2_out[103]), .QB109(buff2_out[109]), .QB110(buff2_out[110]), 32 .QB111(buff2_out[11]), // Read Data
31 32	QB44(buff2_out[44]), QB45(buff2_out[45]), QB46(buff2_out[46]), QB47(buff2_out[47]), // Read Data	52 .Quit(buil_out[1]), // read data
	Page 25 of 84 Ex. 2113 - export_buffers.v	Page 26 of 84 Ex. 2113 - export_buffers.v
1 2 3 4 4 5 5 6 6 7 8 8 9 10 11 12 13 13 14 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1	.QB112(buff2_out[112]),	DA32(ipixel_data2[32]),
32	.DA31(ipixel_data2[31]), // Write Data Page 27 of 84	Page 28 of 84 Ex. 2113 across buffers v.
	Ex. 2113 - export_buffers.v	Ex. 2113 - export_buffers.v

1 2 .i	.DA96(ipixel_data2[96]), .DA97(ipixel_data2[97]), .DA99(ipixel_data2[99]), // Write Data	.98(ipixel_data2[98]),	1 2 .	.TDA16(ipixel_data2[16]), .TDA17(ipixel_data2[17]), .TDA19(ipixel_data2[19]), // Write Test Data	.TDA18(ipixel_data2[18]),
3		12(ipixel_data2[102]),	3	.TDA20(ipixel_data2[20]), .TDA21(ipixel_data2[21]), .TDA23(ipixel_data2[23]), // Write Test Data	.TDA22(ipixel_data2[22]),
5		06(ipixel_data2[106]),	5	.TDA24(ipixel_data2[24]), .TDA25(ipixel_data2[25]), .TDA27(ipixel_data2[27]), // Write Test Data	.TDA26(ipixel_data2[26]),
7	.DA108(ipixel data2[108]), .DA109(ipixel data2[109]), .DA11	0(ipixel_data2[110]),	7	.TDA28(ipixel_data2[28]), .TDA29(ipixel_data2[29]),	.TDA30(ipixel_data2[30]),
9		4(ipixel_data2[114]),	9	.TDA31(ipixel_data2[31]), // Write Test Data .TDA32(ipixel_data2[32]), .TDA33(ipixel_data2[33]),	.TDA34(ipixel_data2[34]),
11	.DA115(ipixel_data2[115]), // Write Data .DA116(ipixel_data2[116]),DA117(ipixel_data2[117]),DA115	8(ipixel_data2[118]),	11	.TDA35(ipixel_data2[35]), // Write Test Data .TDA36(ipixel_data2[36]), .TDA37(ipixel_data2[37]),	.TDA38(ipixel_data2[38]),
13	.DA119(ipixel_data2[119]), // Write Data .DA120(ipixel_data2[120]),DA121(ipixel_data2[121]),DA12	!2(ipixel_data2[122]),	13	.TDA39(ipixel_data2[39]), // Write Test Data .TDA40(ipixel_data2[40]), .TDA41(ipixel_data2[41]),	.TDA42(ipixel_data2[42]),
14 .i	.DA123(ipixel_data2[123]), // Write Data .DA124(ipixel_data2[124]), .DA125(ipixel_data2[125]), .DA12	!6(ipixel_data2[126]),	15	.TDA43(ipixel_data2[43]), // Write Test Data .TDA44(ipixel_data2[44]),TDA45(ipixel_data2[45]),	.TDA46(ipixel_data2[46]),
	.DA127(ipixel_data2[127]), // Write Data // WRITE TEST SIGNALS		16 . 17	.TDA47(ipixel_data2[47]), // Write Test Data .TDA48(ipixel_data2[48]), .TDA49(ipixel_data2[49]),	.TDA50(ipixel_data2[50]),
18	.BISTEA(vss),			.TDA51(ipixel_data2[51]), // Write Test Data .TDA52(ipixel_data2[52]), .TDA53(ipixel_data2[53]),	.TDA54(ipixel_data2[54]),
19 20	.TWEA(vss), // Test write enable .TMEA(vss), // Test memory enable		20	.TDA55(ipixel_data2[55]), // Write Test Data	
21	.TADRA0(iwrite addr[0]), .TADRA1(iwrite addr[1]), .TAD	RA2(iwrite_addr[2]),		.TDA56(ipixel_data2[56]), .TDA57(ipixel_data2[57]), .TDA59(ipixel_data2[59]), // Write Test Data	.TDA58(ipixel_data2[58]),
23	.TADRA3(iwrite_addr[3]), // Write Test Address .TADRA4(iwrite_addr[4]), .TADRA5(iwrite_addr[5]), .TADRA6(iwrite_addr[5]),	ite_addr[6]), // Write		.TDA60(ipixel_data2[60]), .TDA61(ipixel_data2[61]), .TDA63(ipixel_data2[63]), // Write Test Data	.TDA62(ipixel_data2[62]),
25		DA2(ipixel_data2[2]),		.TDA64(ipixel_data2[64]), .TDA65(ipixel_data2[65]), .TDA67(ipixel_data2[67]), // Write Test Data	.TDA66(ipixel_data2[66]),
27	.TDA3(ipixel_data2[3]), // Write Test Data .TDA4(ipixel_data2[4]), .TDA5(ipixel_data2[5]), .TD	DA6(ipixel_data2[6]),	27 28	.TDA68(ipixel_data2[68]), .TDA69(ipixel_data2[69]), .TDA71(ipixel_data2[71]), // Write Test Data	.TDA70(ipixel_data2[70]),
28	.TDA7(ipixel_data2[7]), // Write Test Data	.10(ipixel data2[10]),	29 30	.TDA72(ipixel_data2[72]), .TDA73(ipixel_data2[73]), .TDA75(ipixel_data2[75]), // Write Test Data	.TDA74(ipixel_data2[74]),
	.TDA11(ipixel_data2[11]), // Write Test Data	.14(ipixel_data2[14]),	31 32	.TDA76(ipixel_data2[76]), .TDA77(ipixel_data2[77]), .TDA79(ipixel_data2[79]), // Write Test Data	.TDA78(ipixel_data2[78]),
32 .	TDA15(ipixel_data2[15]), // Write Test Data Page 29 of 84			Page 30 of 84	
		13 - export_buffers.v		1 age 30 01 04	Ex. 2113 - export_buffers.v
3 4 5 6 7 8 5 6 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7 8 7	TDA83(ipixel_data2[83]), Write Test Data TDA84(ipixel_data2[84]), TDA85(ipixel_data2[85]), TDA70A87(ipixel_data2[87]), Write Test Data TDA88(ipixel_data2[87]), Write Test Data TDA88(ipixel_data2[87]), TDA91(ipixel_data2[91]), TDA91(ipixel_data2[91]), TDA91(ipixel_data2[91]), TDA93(ipixel_data2[93]), TDA95(ipixel_data2[95]), Write Test Data TDA96(ipixel_data2[96]), TDA97(ipixel_data2[97]), TDA97(ipixel_data2[97]), TDA996(ipixel_data2[96]), TDA97(ipixel_data2[97]), TDA996(ipixel_data2[96]), TDA97(ipixel_data2[101]), TDA103(ipixel_data2[103]), Write Test Data TDA103(ipixel_data2[103]), Write Test Data TDA104(ipixel_data2[103]), Write Test Data TDA104(ipixel_data2[107]), Write Test Data TDA118(ipixel_data2[117]), Write Test Data TDA112(ipixel_data2[117]), Write Test Data TDA112(ipixel_data2[117]), TDA113(ipixel_data2[117]), TDA11411(ipixel_data2[117]), Write Test Data TDA116(ipixel_data2[115]), Write Test Data TDA116(ipixel_data2[116]), TDA117(ipixel_data2[117]), TDA118(ipixel_data2[121]), TDA119(ipixel_data2[123]), TDA121019(ipixel_data2[123]), Write Test Data TDA120(ipixel_data2[123]), Write Test Data TDA120(ipixel_data2[123]), Write Test Data TDA124(ipixel_data2[123]), Write Test Data TDA124(ipixel_data2[123]), Write Test Data TDA124(ipixel_data2[123]), Write Test Data Write Test Data TDA124(ipixel_data2[123]), Write Test Data TDA124(ipixel_data2[123]), Write Test Data Write Test Data Write Test Data Write Test Data Write Test Data Write Test Data TDA126(ipixel_data2[123]), Write Test Data TDA126(ipixel_data2[123]), Write Test Data Write Test Data Write Wri	16(ipixel_data2[106]), 0(ipixel_data2[110]), 4(ipixel_data2[114]), 8(ipixel_data2[118]), 12(ipixel_data2[122]), 16(ipixel_data2[126]),	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	.AWTB(vss)); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q2_read_mem_addr; assign q2_read_mem_addr = q2_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff3(iRCLK(sclk),	
2 : 3	TDA83(ipixel_data2[83]), // Write Test Data TDA84(ipixel_data2[84]), // TDA85(ipixel_data2[85]), TDA TDA87(ipixel_data2[84]), // TDA85(ipixel_data2[85]), TDA TDA87(ipixel_data2[88]), TDA89(ipixel_data2[89]), TDA TDA91(ipixel_data2[98]), TDA93(ipixel_data2[93]), TDA TDA92(ipixel_data2[95]), // Write Test Data TDA92(ipixel_data2[95]), // Write Test Data TDA96(ipixel_data2[96]), TDA97(ipixel_data2[97]), TDA TDA96(ipixel_data2[96]), TDA97(ipixel_data2[97]), TDA TDA90(ipixel_data2[100]), TDA101(ipixel_data2[101]), TDA10 TDA103(ipixel_data2[104]), TDA105(ipixel_data2[105]), TDA10 TDA104(ipixel_data2[107]), // Write Test Data TDA104(ipixel_data2[107]), // Write Test Data TDA108(ipixel_data2[108]), TDA109(ipixel_data2[109]), TDA11 TDA112(ipixel_data2[115]), TDA113(ipixel_data2[113]), TDA115(ipixel_data2[115]), TDA115(ipixel_data2[115]), TDA116(ipixel_data2[116]), TDA117(ipixel_data2[117]), TDA11 TDA116(ipixel_data2[116]), TDA117(ipixel_data2[117]), TDA11 TDA116(ipixel_data2[116]), TDA117(ipixel_data2[117]), TDA11 TDA119(ipixel_data2[116]), TDA117(ipixel_data2[117]), TDA112 TDA1124(ipixel_data2[120]), TDA125(ipixel_data2[121]), TDA12 TDA124(ipixel_data2[123]), Write Test Data TDA124(ipixel_data2[123]), TDA125(ipixel_data2[125]), TDA12 TDA127(ipixel_data2[127]), Write Test Data TDA129(ipixel_data2[123]), TDA125(ipixel_data2[125]), TDA12 TDA127(ipixel_data2[127]), Write Test Data TDA129(ipixel_data2[123]), TDA125(ipixel_data2[125]), TDA12 TDA129(ipixel_data2[123]), TDA125(ipixel_data2[125]), TDA12 TDA129(ipixel_data2[123]), TDA125(ipixel_data2[125]), TDA12 TDA129(ipixel_data2[123]), TDA125(ipixel_data2[125]), TDA125(ipixel_data2[125]), TDA125(ipixel_data2[125]), TDA125(ipixel_data2[125]), TDA125(ipixel_data2[125]), TDA125(ipixel_da	86(ipixel_data2[86]), 90(ipixel_data2[90]), 94(ipixel_data2[91]), 98(ipixel_data2[98]), 12(ipixel_data2[102]), 16(ipixel_data2[106]), 0(ipixel_data2[110]), 4(ipixel_data2[114]), 8(ipixel_data2[118]), 12(ipixel_data2[122]), 16(ipixel_data2[126]), 16(ipixel_data2[126]),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25); 'endif // !'ifdef USE_BEHAVE_MEM wire [6:0] q2_read_mem_addr; assign q2_read_mem_addr = q2_read_addr[6:0]; 'ifdef USE_BEHAVE_MEM dum_mem_p2 #(7,128) bank0_buff3(.iRCLK(sclk), .iWCLK(sclk), .iMER(q2_mem_re), .iMEW(imem_wen), .iWEN(imem_wew), .iRADR(q2_read_addr[6:0]), .iWADR(iwrite_addr), .iD(ipixel_data3), .oQ(buff3_out));	t iwrite_addr q2_read_addr

1	, .QB3(buff3_out[3]), // Read , .QB7(buff3_out[7]), // Read	1
15	.QB14(buff3_out[14]), QB18(buff3_out[22]), .QB22(buff3_out[22]), .QB26(buff3_out[26]), .QB30(buff3_out[30]), .QB34(buff3_out[34]), .QB38(buff3_out[34]), .QB42(buff3_out[42]), .QB46(buff3_out[46]), Ex. 2113 - export_buffers.v	17 QB80(buff3_out[80]), QB81(buff3_out[81]), QB82(buff3_out[82]), 18 QB83(buff3_out[83]), // Read Data 19 QB84(buff3_out[84]), QB85(buff3_out[85]), 20 QB87(buff3_out[87]), // Read Data 21 QB88(buff3_out[88]), QB89(buff3_out[89]), 22 QB91(buff3_out[91]), // Read Data 23 QB92(buff3_out[92]), // Read Data 24 QB95(buff3_out[95]), // Read Data 25 QB96(buff3_out[96]), // Read Data 26 QB99(buff3_out[96]), // Read Data 27 QB100(buff3_out[100]), QB101(buff3_out[101]), QB108(buff3_out[102]), 28 QB103(buff3_out[103]), // Read Data 29 QB104(buff3_out[104]), QB105(buff3_out[105]), QB106(buff3_out[106]), 30 QB107(buff3_out[107]), // Read Data 31 QB108(buff3_out[108]), QB109(buff3_out[109]), 32 QB11(buff3_out[107]), // Read Data Page 34 of 84 Ex. 2113 - export_buffers.v
1 QB112(buff3_out[112]), QB113(buff3_out[113]), 2 QB115(buff3_out[115]), // Read Data 3 QB116(buff3_out[116]), // Read Data 5 QB120(buff3_out[120]), // Read Data 5 QB123(buff3_out[120]), // Read Data 7 QB123(buff3_out[123]), // Read Data 7 QB124(buff3_out[123]), // Read Data 9 // WRITE INTERFACE 10 CLKA(iSCLK), // Write Clock 11 WEA(imem_wen), // Write enable 12 MEA(vdd), // Memory enable 13 ADRA0(iwrite_addr[0]), ./ ADRA1(iwrite_addr[1]), 14 ADRA3(iwrite_addr[4]), ./ ADRA5(iwrite_addr[5]), ./ ADRA1(iwrite_addr[5]), ./ Write Data 15 ADRA4(iwrite_addr[4]), ./ DAS(ipixel_data3[1]), 18 DA3(ipixel_data3[0]), // Write Data 19 DA4(ipixel_data3[1]), // Write Data 19 DA4(ipixel_data3[1]), // Write Data 21 DA8(ipixel_data3[1]), // Write Data 22 DA11(ipixel_data3[1]), // Write Data 23 DA12(ipixel_data3[12]), ./ DA15(ipixel_data3[13]), 24 DA15(ipixel_data3[15]), // Write Data 25 DA16(ipixel_data3[15]), // Write Data 26 DA19(ipixel_data3[15]), // Write Data 27 DA20(ipixel_data3[16]), .// DA17(ipixel_data3[17]), 28 DA23(ipixel_data3[20]), // Write Data 29 DA24(ipixel_data3[20]), // Write Data 21 DA29(ipixel_data3[16]), .// DA17(ipixel_data3[17]), 22 DA17(ipixel_data3[17]), // Write Data 23 DA19(ipixel_data3[18]), // Write Data 24 DA15(ipixel_data3[18]), // Write Data 25 DA16(ipixel_data3[18]), // Write Data 26 DA24(ipixel_data3[20]), // Write Data 27 DA20(ipixel_data3[20]), // Write Data 28 DA23(ipixel_data3[20]), // Write Data 31 DA28(ipixel_data3[21]), // Write Data	.QB114(buff3_out[114]), .QB118(buff3_out[118]), .QB122(buff3_out[122]), .QB126(buff3_out[126]), .QB126(buff3_out[126]), .ADRA2(iwrite_addr[2]), .6(iwrite_addr[6]), // Write .DA2(ipixel_data3[2]), .DA6(ipixel_data3[10]), .DA14(ipixel_data3[14]), .DA18(ipixel_data3[14]), .DA22(ipixel_data3[12]), .DA26(ipixel_data3[26]), .DA26(ipixel_data3[26]), .DA30(ipixel_data3[30]),	1
Page 35 of 84	Ex. 2113 - export_buffers.v	Page 36 of 84 Ex. 2113 - export_buffers.v

1 2	.DA96(ipixel_data3[96]), .DA97(ipixel_data3[97]), .DA98(ipixel_data3[98]), .DA99(ipixel_data3[98]),	1 .TDA16(ipixel_data3[16]), .TDA17(ipixel_data3[17]), .TDA18(ipixel_data3[18]), 2 .TDA19(ipixel_data3[19]), // Write Test Data
3 4	.DA100(ipixel_data3[100]), .DA101(ipixel_data3[101]), .DA102(ipixel_data3[102]), .DA103(ipixel_data3[103]), // Write Data	3 .TDA20(ipixel_data3[20]), .TDA21(ipixel_data3[21]), .TDA22(ipixel_data3[22]), 4 .TDA23(ipixel_data3[23]), // Write Test Data
5	.DA104(ipixel_data3[104]), .DA105(ipixel_data3[105]), .DA106(ipixel_data3[106]), .DA107(ipixel_data3[107]), // Write Data	5 .TDA24(ipixel_data3[24]), .TDA25(ipixel_data3[25]), .TDA26(ipixel_data3[26]), 6 .TDA27(ipixel_data3[27]), // Write Test Data
7 8	DA108(ipixel_data3[108]),DA109(ipixel_data3[109]),DA110(ipixel_data3[110]),DA111(ipixel_data3[111]),DA111(ipixel_	7TDA28(ipixel_data3[28]),TDA29(ipixel_data3[29]),TDA30(ipixel_data3[30]), 8TDA31(ipixel_data3[31]), // Write Test Data
9	.DA112(ipixel_data3[112]), .DA113(ipixel_data3[113]), .DA114(ipixel_data3[114]), .DA115(ipixel_data3[115]), // Write Data	9TDA32(ipixel_data3[32]),TDA33(ipixel_data3[33]),TDA34(ipixel_data3[34]), 10TDA35(ipixel_data3[35]), // Write Test Data
11 12	.DA116(ipixel_data3[116]), .DA117(ipixel_data3[117]), .DA118(ipixel_data3[118]), .DA119(ipixel_data3[118]),DA11	11TDA36(ipixel_data3[36]),TDA37(ipixel_data3[37]),TDA38(ipixel_data3[38]), 12TDA39(ipixel_data3[39]),Write Test Data
13 14	.DA120(ipixel_data3[123]), // Mrite Data .DA121(ipixel_data3[121]), .DA122(ipixel_data3[122]), .DA123(ipixel_data3[123]), // Write Data	13TDA40(ipixel_data3[40]),TDA41(ipixel_data3[41]),TDA42(ipixel_data3[42]), 14TDA43(ipixel_data3[43]), // Write Test Data
15 16	.DA124(ipixel_data3[124]), .DA125(ipixel_data3[125]), .DA126(ipixel_data3[126]), .DA127(ipixel_data3[127]), // Write Data	15TDA44(ipixel_data3[44]),TDA45(ipixel_data3[45]),TDA46(ipixel_data3[46]), 16TDA47(ipixel_data3[47]), // Write Test Data
17	// WRITE TEST SIGNALS	17TDA48(ipixel_data3[48]),TDA49(ipixel_data3[49]),TDA50(ipixel_data3[50]), 18TDA51(ipixel_data3[51]), // Write Test Data
18 19	.BISTEA(vss), .TWEA(vss), // Test write enable	19TDA52(ipixel_data3[52]),TDA53(ipixel_data3[53]),TDA54(ipixel_data3[54]), 20TDA55(ipixel_data3[55]),Write Test Data
20	.TMEA(vss), // Test memory enable	20TDA56(ipixel_data3[56]), // write Test Data 21TDA56(ipixel_data3[56]),TDA57(ipixel_data3[57]),TDA58(ipixel_data3[58]), 22TDA59(ipixel_data3[59]), // Write Test Data
21 22	$. TADRA0(iwrite_addr[0]), \qquad . TADRA1(iwrite_addr[1]), \qquad . TADRA2(iwrite_addr[2]), \\ . TADRA3(iwrite_addr[3]), \ // Write Test Address$	22TDA50(ipixel_data3[60]), // write Test Data 23TDA60(ipixel_data3[60]),TDA61(ipixel_data3[61]),TDA62(ipixel_data3[62]), 24TDA63(ipixel_data3[63]), // Write Test Data
23 24	$. TADRA4(iwrite_addr[4]), \ TADRA5(iwrite_addr[5]), \ TADRA6(iwrite_addr[6]), \ \#Write \ Test \ Address$	24 . IDAO5(ipixel_data5[64]), // write Test Data 25TDA65(ipixel_data3[64]), .TDA65(ipixel_data3[65]), .TDA66(ipixel_data3[66]), 26TDA67(ipixel_data3[67]), // Write Test Data
25 26	$. TDA0(ipixel_data3[0]), \qquad . TDA1(ipixel_data3[1]), \qquad . TDA2(ipixel_data3[2]), \\ . TDA3(ipixel_data3[3]), \ // Write Test Data$	26 . IDAO((ipixel_data3[68]), // witte Test Data 27TDA68(ipixel_data3[68]),TDA69(ipixel_data3[69]),TDA70(ipixel_data3[70]), 28TDA71(ipixel_data3[71]), // Write Test Data
27 28	.TDA4(ipixel_data3[4]), .TDA5(ipixel_data3[5]), .TDA6(ipixel_data3[6]), .TDA7(ipixel_data3[7]), // Write Test Data	29TDA72(ipixel_data3[72]),TDA73(ipixel_data3[73]),TDA74(ipixel_data3[74]), 30TDA75(ipixel_data3[75]),Write Test Data
29 30	.TDA8(ipixel_data3[8]), .TDA9(ipixel_data3[9]), .TDA10(ipixel_data3[10]), .TDA11(ipixel_data3[11]), // Write Test Data	31TDA76(ipixel_data3[76]), "Write Yest Data 32TDA79(ipixel_data3[79]), "Write Yest Data 33TDA79(ipixel_data3[79]), "Write Yest Data
31 32	.TDA12(ipixel_data3[12]), .TDA13(ipixel_data3[13]), .TDA14(ipixel_data3[14]), .TDA15(ipixel_data3[15]), // Write Test Data	22
	Page 37 of 84 Ex. 2113 - export_buffers.v	Page 38 of 84 Ex. 2113 - export_buffers.v
1 2 3 4 4 5 5 6 6 7 7 8 9 9 10 11 112 12 13 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28	.TDA80(ipixel_data3[80]), .TDA81(ipixel_data3[81]), .TDA82(ipixel_data3[82]), .TDA83(ipixel_data3[83]), .// Write Test Data .TDA84(ipixel_data3[84]),	1 .AWTB(vss) 2); 3 4 'endif // Vifdef USE_BEHAVE_MEM 5 6 7 'ifdef USE_BEHAVE_MEM 8 dum_mem_p2 #(7,128) bank1_buff0(.iRCLK(sclk), 9 .iWCLK(sclk), 10 .iMER(imem_re), 11 .iMEW(imem_wen), 12 .iWEN(imem_wen), 13 .iRADR(iread_addr[6-0]), 14 .iWADR(iwrite_addr), 15 .iD(ipixel_data4), 16 .oQ(buff4_out) 17 .); 18 'else // Fifdef USE_BEHAVE_MEM 19 20 rfsd2_80x128cm2sw0 ubank1_buff0 21 .(*VRGIO rfsd2_80x128cm2sw0 ipixel_data4 buff4_out iwrite_addr iread_addr imem_wen imem_re null*/ 22 imem_re null*/
29 30	$. TADRB0(q2_read_addr[0]), . TADRB1(q2_read_addr[1]), . TADRB2(q2_read_addr[2]), \\ . TADRB3(q2_read_addr[3]), \ \# Read Test Address$	24 .CLKB(iSCLK), // Read Clock 25 .OEB(vdd), // Output enable
31 32	$. TADRB4(q2_read_addr[4]), . TADRB5(q2_read_addr[5]), . TADRB6(q2_read_addr[6]), \ // Read Test Address$	26 .MEB(imem_re), // Read enable
	Page 39 of 84 Ex. 2113 - export_buffers.v	Page 40 of 84 Ex. 2113 - export_buffers.v

 ADRB0(iread_addr[0]), ADRB1(iread_addr[1]), ADRB2(iread_addr[2]), ADRB3(iread_addr[3]), // Read Address 	1 .QB56(buff4_out[56]), .QB57(buff4_out[57]), .QB58(buff4_out[58]), 2 .QB59(buff4_out[59]), // Read Data
3 .ADRB4(iread_addr[4]), .ADRB5(iread_addr[5]), .ADRB6(iread_addr[6]), // Read 4 Address	3 .QB60(buff4_out[60]), .QB61(buff4_out[61]), .QB62(buff4_out[62]), 4 .QB63(buff4_out[63]), // Read Data
$\label{eq:continuous} \begin{array}{lll} 5 & & .QB0(buff4_out[0]), .QB1(buff4_out[1]), .QB2(buff4_out[2]), .QB3(buff4_out[3]), \ // \ Read \\ 6 & Data \\ \end{array}$	5 .QB64(buff4_out[64]), .QB65(buff4_out[65]), .QB66(buff4_out[66]), 6 .QB67(buff4_out[67]), // Read Data
7QB4(buff4_out[4]), .QB5(buff4_out[5]), .QB6(buff4_out[6]), .QB7(buff4_out[7]), // Read 8Data	7 .QB68(buff4_out[68]), .QB69(buff4_out[69]), .QB70(buff4_out[70]), 8 .QB71(buff4_out[71]), // Read Data
9QB8(buff4_out[8]), .QB9(buff4_out[9]), .QB10(buff4_out[10]), .QB11(buff4_out[11]), // 10 Read Data	9 .QB72(buff4_out[72]), .QB73(buff4_out[73]), .QB74(buff4_out[74]), 10 .QB75(buff4_out[75]), // Read Data
11 .QB12(buff4_out[12]), .QB13(buff4_out[13]), .QB14(buff4_out[14]), 12 .QB15(buff4_out[15]), // Read Data	11 .QB76(buff4_out[76]), .QB77(buff4_out[77]), .QB78(buff4_out[78]),
13 .QB16(buff4_out[16]), .QB17(buff4_out[17]), .QB18(buff4_out[18]), 14 .QB19(buff4_out[19]), // Read Data	13 .QB80(buff4_out[80]), .QB81(buff4_out[81]), .QB82(buff4_out[82]), 14 .QB83(buff4_out[83]), // Read Data
15 .QB20(buff4_out[20]), .QB21(buff4_out[21]), .QB22(buff4_out[22]), 16 .QB23(buff4_out[23]), // Read Data	15 .QB84(buff4_out[84]), .QB85(buff4_out[85]), .QB86(buff4_out[86]), .QB87(buff4_out[87]), // Read Data
17 .QB24(buff4_out[24]), .QB25(buff4_out[25]), .QB26(buff4_out[26]), 18 .QB27(buff4_out[27]), // Read Data	17 .QB88(buff4_out[88]), .QB89(buff4_out[89]), .QB90(buff4_out[90]), .QB90(buff4_out[91]), // Read Data
19 .QB28(buff4_out[28]), .QB29(buff4_out[29]), .QB30(buff4_out[30]), 20 .QB31(buff4_out[31]), // Read Data	19 .QB92(buff4_out[92]), .QB93(buff4_out[93]), .QB94(buff4_out[94]), .QB95(buff4_out[94]), .QB95(buff4_out[95]), // Read Data
21 .QB32(buff4_out[32]), .QB33(buff4_out[33]), .QB34(buff4_out[34]), 22 .QB35(buff4_out[35]), // Read Data	21 .QB96(buff4_out[96]), .QB97(buff4_out[97]), .QB98(buff4_out[98]), .22 .QB99(buff4_out[99]), // Read Data
23 .QB36(buff4_out[36]), .QB37(buff4_out[37]), .QB38(buff4_out[38]), 24 .QB39(buff4_out[39]), // Read Data	23 .QB100(buff4_out[100]), .QB101(buff4_out[101]), .QB102(buff4_out[102]), .QB103(buff4_out[103]), // Read Data
25 .QB40(buff4_out[40]), .QB41(buff4_out[41]), .QB42(buff4_out[42]),	25 .QB104(buff4_out[104]), .QB105(buff4_out[105]), .QB106(buff4_out[106]),
26 QB43(buff4_out[43]), // Read Data 27 QB44(buff4_out[44]), QB45(buff4_out[45]), QB46(buff4_out[46]),	26 .QB107(buff4_out[107]), // Read Data 27 .QB108(buff4_out[108]), .QB109(buff4_out[109]), .QB110(buff4_out[110]),
28 .QB47(buff4_out[47]), // Read Data 29 .QB48(buff4_out[48]), .QB49(buff4_out[49]), .QB50(buff4_out[50]),	28 .QB111(buff4_out[111]), // Read Data 29 .QB112(buff4_out[112]),QB113(buff4_out[113]),QB114(buff4_out[114]),
30 .QB51(buff4_out[51]), // Read Data 31 .QB52(buff4_out[52]), .QB53(buff4_out[53]), .QB54(buff4_out[54]),	30 .QB115(buff4_out[115]), // Read Data 31 .QB116(buff4_out[116]), .QB117(buff4_out[117]), .QB118(buff4_out[118]),
32 .QB55(buff4_out[55]), // Read Data	32 .QB119(buff4_out[119]), // Read Data
Page 41 of 84 Ex. 2113 - export_buffers.v	Page 42 of 84 Ex. 2113 - export_buffers.v
OB120(buff4_out[120]),	1
29 .DA32(ipixel_data4[32]), .DA33(ipixel_data4[33]), .DA34(ipixel_data4[34]), 30 .DA35(ipixel_data4[35]), // Write Data	30 .DA99(ipixel_data4[99]), // Write Data 31 .DA100(ipixel_data4[100]), .DA101(ipixel_data4[101]), .DA102(ipixel_data4[102]), 32 .DA102(ipixel_data4[102]), // Write Data
31 .DA36(ipixel_data4[36]), .DA37(ipixel_data4[37]), .DA38(ipixel_data4[38]), 32 .DA39(ipixel_data4[39]), // Write Data	32 .DA103(ipixel_data4[103]), // Write Data
Page 43 of 84 Ex. 2113 - export_buffers.v	Page 44 of 84 Ex. 2113 - export_buffers.y

1	.DA104(ipixel_data4[104]),DA105(ipixel_data4[105]),DA106(ipixel_data4[106]),	1 .TDA24(ipixel_data4[24]), .TDA25(ipixel_data4[25]), .TDA26(ipixel_data4[26]),
2	.DA107(ipixel_data4[107]), // Write Data .DA108(ipixel_data4[108]), .DA109(ipixel_data4[109]), .DA110(ipixel_data4[110]),	2 .TDA27(ipixel_data4[27]), // Write Test Data 3 .TDA28(ipixel_data4[28]), .TDA29(ipixel_data4[29]), .TDA30(ipixel_data4[30]),
4	.DA111(ipixel_data4[111]), // Write Data .DA112(ipixel_data4[112]), .DA113(ipixel_data4[113]), .DA114(ipixel_data4[114]),	4 .TDA31(ipixel_data4[31]), // Write Test Data 5 .TDA32(ipixel_data4[32]), .TDA33(ipixel_data4[33]), .TDA34(ipixel_data4[34]),
6	.DA115(ipixel_data4[115]), // Write Data	6 .TDA35(ipixel_data4[35]), // Write Test Data 7 .TDA36(ipixel_data4[36]), .TDA37(ipixel_data4[37]), .TDA38(ipixel_data4[38]),
8	.DA119(ipixel_data4[119]), // Write Data	8 .TDA39(ipixel_data4[39]), // Write Test Data
9 10	.DA120(ipixel_data4[120]), .DA121(ipixel_data4[121]), .DA122(ipixel_data4[122]), .DA123(ipixel_data4[123]), // Write Data	9 .TDA40(ipixel_data4[40]), .TDA41(ipixel_data4[41]), .TDA42(ipixel_data4[42]), 10 .TDA43(ipixel_data4[43]), // Write Test Data
11 12	.DA124(ipixel_data4[124]), .DA125(ipixel_data4[125]), .DA126(ipixel_data4[126]), .DA127(ipixel_data4[127]), // Write Data	11 .TDA44(ipixel_data4[44]), .TDA45(ipixel_data4[45]), .TDA46(ipixel_data4[46]), 12 .TDA47(ipixel_data4[47]), // Write Test Data
13	// WRITE TEST SIGNALS	13 .TDA48(ipixel_data4[48]), .TDA9(ipixel_data4[49]), .TDA50(ipixel_data4[50]), 14 .TDA51(ipixel_data4[51]), // Write Test Data
14 15	.BISTEA(vss), .TWEA(vss), // Test write enable	15 .TDA52(ipixel_data4[52]), .TDA53(ipixel_data4[53]), .TDA54(ipixel_data4[54]), 16 .TDA55(ipixel_data4[55]), // Write Test Data
16	.TMEA(vss), // Test memory enable	17 .TDA56(ipixel_data4[56]), .TDA57(ipixel_data4[57]), .TDA58(ipixel_data4[58]), 18 .TDA59(ipixel_data4[59]), // Write Test Data
17 18	$. TADRA0(iwrite_addr[0]), \qquad . TADRA1(iwrite_addr[1]), \qquad . TADRA2(iwrite_addr[2]), \\ . TADRA3(iwrite_addr[3]), \ // Write\ Test\ Address$	19 .TDA60(ipixel_data4[60]), .TDA61(ipixel_data4[61]), .TDA62(ipixel_data4[62]),
19 20	.TADRA4(iwrite_addr[4]), .TADRA5(iwrite_addr[5]), .TADRA6(iwrite_addr[6]), // Write Test Address	20 .TDA63(ipixel_data4[63]), // Write Test Data 21 .TDA64(ipixel_data4[64]), .TDA65(ipixel_data4[65]), .TDA66(ipixel_data4[66]),
21 22	.TDA0(ipixel_data4[0]), .TDA1(ipixel_data4[1]), .TDA2(ipixel_data4[2]), .TDA3(ipixel_data4[3]), // Write Test Data	22 .TDA67(ipixel_data4[67]), // Write Test Data
23	.TDA5(ipixel_data4[4]), .TDA5(ipixel_data4[5]), .TDA6(ipixel_data4[6]),	24 .TDA71(ipixel_data4[71]), // Write Test Data
24 25	.TDA7(ipixel_data4[7]), // Write Test Data .TDA8(ipixel_data4[8]), .TDA9(ipixel_data4[9]), .TDA10(ipixel_data4[10]),	25TDA72(ipixel_data4[72]),TDA73(ipixel_data4[73]),TDA74(ipixel_data4[74]),TDA75(ipixel_data4[75]), // Write Test Data
26 27	.TDA11(ipixel_data4[11]), // Write Test Data .TDA12(ipixel_data4[12]),TDA13(ipixel_data4[13]),TDA14(ipixel_data4[14]),	27 .TDA76(ipixel_data4[76]), .TDA77(ipixel_data4[77]), .TDA78(ipixel_data4[78]), 28 .TDA79(ipixel_data4[79]), // Write Test Data
28	.TDA15(ipixel_data4[15]), // Write Test Data .TDA16(ipixel_data4[16]),TDA17(ipixel_data4[17]),TDA18(ipixel_data4[18]),	29 .TDA80(ipixel_data4[80]), .TDA81(ipixel_data4[81]), .TDA82(ipixel_data4[82]), 30 .TDA83(ipixel_data4[83]), // Write Test Data
30	.TDA19(ipixel_data4[19]), // Write Test Data	31 .TDA84(ipixel_data4[84]), .TDA85(ipixel_data4[85]), .TDA86(ipixel_data4[86]), 32 .TDA87(ipixel_data4[87]), // Write Test Data
31 32	.TDA20(ipixel_data4[20]), .TDA21(ipixel_data4[21]), .TDA22(ipixel_data4[22]), .TDA23(ipixel_data4[23]), .// Write Test Data	
	Page 45 of 84 Ex. 2113 - export_buffers.v	Page 46 of 84 Ex. 2113 - export_buffers.v
2 3 4 5 6 7 8	.TDA91(ipixel_data4[91]), // Write Test Data .TDA92(ipixel_data4[92]), .TDA93(ipixel_data4[93]), .TDA94(ipixel_data4[94]), .TDA95(ipixel_data4[95]), // Write Test Data .TDA96(ipixel_data4[96]), .TDA97(ipixel_data4[97]), .TDA98(ipixel_data4[98]), // Write Test Data .TDA100(ipixel_data4[100]), .TDA101(ipixel_data4[101]), .TDA102(ipixel_data4[102]), .TDA103(ipixel_data4[103]), // Write Test Data	2 3 'ifdef USE_BEHAVE_MEM 4 dum_mem_p2 #(7,128) bank1_buff1(.iRCLK(sclk), 5 .iWCLK(sclk),
9 10 11 12 13 14 15 16	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), // Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), // Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), // Write Test Data .TDA116(ipixel_data4[116]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), // Write Test Data .TDA110(ipixel_data4[119]), // Write Test Data .TDA120(ipixel_data4[110]), .TDA121(ipixel_data4[111]), .TDA122(ipixel_data4[120]),	5
10 11 12 13 14 15 16	TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), // Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), // Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), // Write Test Data .TDA116(ipixel_data4[116]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), // Write Test Data .TDA120(ipixel_data4[120]), .TDA121(ipixel_data4[121]), .TDA122(ipixel_data4[122]), .TDA123(ipixel_data4[123]), // Write Test Data	6
10 11 12 13 14 15 16 17 18	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), .// Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), .// Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), .TDA115(ipixel_data4[115]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), .TDA119(ipixel_data4[119]), .TDA119(ipixel_data4[121]), .TDA119(ipixel_data4[123]), .TDA120(ipixel_data4[123]), .TDA121(ipixel_data4[121]), .TDA122(ipixel_data4[122]), .TDA124(ipixel_data4[123]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]), .TDA126(ipixel_dat	6
10 11 12 13 14 15 16 17 18	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), ./// Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), ./// Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), ./// Write Test Data .TDA116(ipixel_data4[116]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), ./// Write Test Data .TDA120(ipixel_data4[120]), .TDA121(ipixel_data4[121]), .TDA122(ipixel_data4[122]), .TDA123(ipixel_data4[123]), ./// Write Test Data .TDA124(ipixel_data4[123]), ./// Write Test Data .TDA124(ipixel_data4[123]), ./// Write Test Data .TDA124(ipixel_data4[124]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]),	6
10 11 12 13 14 15 16 17 18	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), // Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), .TDA111(ipixel_data4[111]), .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA120(ipixel_data4[123]), .TDA121(ipixel_data4[121]), .TDA122(ipixel_data4[122]), .TDA123(ipixel_data4[123]), .TDA124(ipixel_data4[124]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]), .TDA127(ipixel_data4[127]), .// Write Test Data .TDA124(ipixel_data4[127]), .// Write Test Data .// READ TEST SIGNALS	6
10 11 12 13 14 15 16 17 18 19 20 21 22 23	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), // Write Test Data .TDA108(ipixel_data4[108]), .TDA108(ipixel_data4[110]), .TDA111(ipixel_data4[111]), .TDA111(ipixel_data4[111]), .TDA113(ipixel_data4[112]), .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA120(ipixel_data4[123]), .TDA120(ipixel_data4[123]), .TDA121(ipixel_data4[123]), .TDA122(ipixel_data4[123]), .TDA122(ipixel_data4[124]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]), .TDA127(ipixel_data4[127]), .TDA125(ipixel_data4[127]), .TDA125(ipixel_data4[128]), .TDA125(ipi	6
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), ./// Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), ./// Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), ./// Write Test Data .TDA116(ipixel_data4[116]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), ./// Write Test Data .TDA120(ipixel_data4[120]), .TDA121(ipixel_data4[121]), .TDA122(ipixel_data4[122]), .TDA123(ipixel_data4[123]), ./// Write Test Data .TDA124(ipixel_data4[124]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]), .TDA127(ipixel_data4[127]), ./// Write Test Data ./// READ TEST SIGNALS .BISTEB(vss), .TOEB(vss), .TOEB(vss), .TADRB0(iread_addr[0]),TADRB1(iread_addr[1]),TADRB2(iread_addr[2]), .TADRB3(iread_addr[3]), ./// Read Test Address	6
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), // Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), // Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), // Write Test Data .TDA116(ipixel_data4[116]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), // Write Test Data .TDA120(ipixel_data4[120]), // Write Test Data .TDA120(ipixel_data4[123]), // Write Test Data .TDA124(ipixel_data4[123]), // Write Test Data .TDA124(ipixel_data4[124]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]), .TDA127(ipixel_data4[127]), // Write Test Data // READ TEST SIGNALS .BISTEB(vss), .TOEB(vss), .TOEB(vss), .TADRB0(iread_addr[0]), .TADRB1(iread_addr[1]), .TADRB2(iread_addr[2]),	6
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	.TDA104(ipixel_data4[104]), .TDA105(ipixel_data4[105]), .TDA106(ipixel_data4[106]), .TDA107(ipixel_data4[107]), // Write Test Data .TDA108(ipixel_data4[108]), .TDA109(ipixel_data4[109]), .TDA110(ipixel_data4[110]), .TDA111(ipixel_data4[111]), // Write Test Data .TDA112(ipixel_data4[112]), .TDA113(ipixel_data4[113]), .TDA114(ipixel_data4[114]), .TDA115(ipixel_data4[115]), // Write Test Data .TDA116(ipixel_data4[115]), // Write Test Data .TDA119(ipixel_data4[119]), .TDA117(ipixel_data4[117]), .TDA118(ipixel_data4[118]), .TDA119(ipixel_data4[119]), .TDA121(ipixel_data4[121]), .TDA122(ipixel_data4[122]), .TDA123(ipixel_data4[123]), .TDA121(ipixel_data4[121]), .TDA123(ipixel_data4[123]), .TDA125(ipixel_data4[125]), .TDA126(ipixel_data4[126]), .TDA127(ipixel_data4[127]), .TDA128(ipixel_data4[127]), .TDA128(ipixel_data4[127]), .TDA128(ipixel_data4[126]), .TDA128(ipixel_data4[127]), .TDA128(ipixel_data4[126]), .TDA128(ip	6
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	TDA104(ipixel_data4[104]), TDA105(ipixel_data4[105]), TDA106(ipixel_data4[106]), TDA107(ipixel_data4[107]), // Write Test Data TDA108(ipixel_data4[108]), TDA109(ipixel_data4[109]), TDA110(ipixel_data4[110]), TDA111(ipixel_data4[111]), // Write Test Data TDA112(ipixel_data4[111]), // Write Test Data TDA112(ipixel_data4[112]), // TDA113(ipixel_data4[113]), TDA114(ipixel_data4[114]), TDA115(ipixel_data4[115]), // Write Test Data TDA116(ipixel_data4[116]), // Write Test Data TDA119(ipixel_data4[119]), // Write Test Data TDA120(ipixel_data4[120]), // TDA121(ipixel_data4[121]), TDA122(ipixel_data4[122]), TDA123(ipixel_data4[123]), // Write Test Data TDA124(ipixel_data4[123]), // Write Test Data // TDA124(ipixel_data4[127]), // Write Test Data // READ TEST SIGNALS .BISTEB(vss), TOEB(vss), TADRB0(iread_addr[0]), // TADRB1(iread_addr[1]), // TADRB2(iread_addr[2]), // TADRB3(iread_addr[4]), // Read Test Address .TADRB4(iread_addr[4]), // TADRB5(iread_addr[5]), // TADRB6(iread_addr[6]), // Read Test Address	6

1 .QB4(buff5_out[4]), .QB5(buff5_out[5]), .QB6(buff5_out[6]), .QB7(buff5_out[7]), // Read 2 Data	1 .QB68(buff5_out[68]), .QB69(buff5_out[69]), .QB70(buff5_out[70]), 2 .QB71(buff5_out[71]), // Read Data
3 .QB8(buff5_out[8]), .QB9(buff5_out[9]), .QB10(buff5_out[10]), .QB11(buff5_out[11]), // 4 Read Data	3 .QB72(buff5_out[72]), .QB73(buff5_out[73]), .QB74(buff5_out[74]), 4 .QB75(buff5_out[75]), // Read Data
5 .QB12(buff5_out[12]), .QB13(buff5_out[13]), .QB14(buff5_out[14]), 6 .QB15(buff5_out[15]), // Read Data	5 .QB76(buff5_out[76]), .QB77(buff5_out[77]), .QB78(buff5_out[78]), 6 .QB79(buff5_out[79]), // Read Data
7 .QB16(buff5_out[16]), .QB17(buff5_out[17]), .QB18(buff5_out[18]), 8 .QB19(buff5_out[19]), // Read Data	7 .QB80(buff5_out[80]), .QB81(buff5_out[81]), .QB82(buff5_out[82]), 8 .QB83(buff5_out[83]), // Read Data
9 .QB20(buff5_out[20]), .QB21(buff5_out[21]), .QB22(buff5_out[22]), 10 .QB23(buff5_out[23]), // Read Data	9 .QB84(buff5_out[84]), .QB85(buff5_out[85]), .QB86(buff5_out[86]), 10 .QB87(buff5_out[87]), // Read Data
11	11 .QB88(buff5_out[88]), .QB89(buff5_out[89]), .QB90(buff5_out[90]), 12 .QB91(buff5_out[91]), // Read Data
13	13 .QB92(buff5_out[92]), .QB93(buff5_out[93]), .QB94(buff5_out[94]), .14 .QB95(buff5_out[95]), // Read Data
15	15 .QB96(buff5_out[96]), .QB97(buff5_out[97]), .QB98(buff5_out[98]),
17	17 .QB100(buff5_out[100]), .QB101(buff5_out[101]), .QB102(buff5_out[102]), 18 .QB103(buff5_out[103]), // Read Data
19 .QB40(buff5_out[40]), .QB41(buff5_out[41]), .QB42(buff5_out[42]), .QB43(buff5_out[43]), // Read Data	19 .QB104(buff5_out[104]), .QB105(buff5_out[105]), .QB106(buff5_out[106]), 20 .QB107(buff5_out[107]), // Read Data
21 .QB44(buff5_out[44]), .QB45(buff5_out[45]), .QB46(buff5_out[46]), .22 .QB47(buff5_out[47]), // Read Data	21 .QB108(buff5_out[108]), .QB109(buff5_out[109]), .QB110(buff5_out[110]), 22 .QB111(buff5_out[11]), // Read Data
23 .QB48(buff5_out[48]), .QB49(buff5_out[49]), .QB50(buff5_out[50]), .QB50(buff5_out[51]), // Read Data	23 .QB112(buff5_out[112]), .QB113(buff5_out[113]), .QB114(buff5_out[114]), 24 .QB115(buff5_out[115]), // Read Data
25 .QB52(buff5_out[52]), .QB53(buff5_out[53]), .QB54(buff5_out[54]), .QB55(buff5_out[55]), // Read Data	25 .QB116(btff5_out[116]), .QB117(btff5_out[117]), .QB118(btff5_out[118]), 26 .QB119(btff5_out[119]), // Read Data
27 .QB56(buff5_out[56]), .QB57(buff5_out[57]), .QB58(buff5_out[58]), .QB59(buff5_out[59]), // Read Data	27QB120(buff5_out[120]), _QB121(buff5_out[121]), _QB122(buff5_out[122]), _28QB123(buff5_out[123]), // Read Data
29 .QB60(buff5_out[60]), .QB61(buff5_out[61]), .QB62(buff5_out[62]), 30 .QB63(buff5_out[63]), // Read Data	29 .QB124(buff5_out[124]), .QB125(buff5_out[125]), .QB126(buff5_out[126]), 30 .QB127(buff5_out[127]), // Read Data
31 .QB64(buff5_out[64]), .QB65(buff5_out[65]), .QB66(buff5_out[66]), 32 .QB67(buff5_out[67]), // Read Data	31 // WRITE INTERFACE
	32 .CLKA(iSCLK), // Write Clock
Page 49 of 84	Decc 50 of 94
Ex. 2113 - export_buffers.v	Page 50 of 84 Ex. 2113 - export_buffers.v
Ex. 2113 - export_buffers.v 1 .WEA(imem_wen), // Write enable	Ex. 2113 - export_buffers.v 1 .DA52(ipixel_data5[52]), .DA53(ipixel_data5[53]), .DA54(ipixel_data5[54]),
Ex. 2113 - export_buffers.v	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1 .WEA(imem_wen), // Write enable 2 .MEA(vdd), // Memory enable	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1 .WEA(imem_wen), // Write enable 2 .MEA(vdd), // Memory enable 3 .ADRA0(iwrite_addr[0]), .ADRA1(iwrite_addr[1]), .ADRA2(iwrite_addr[2]), 4 .ADRA3(iwrite_addr[3]), // Write Address	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1 .WEA(imem_wen), // Write enable 2 .MEA(vdd), // Memory enable 3 .ADRA0(iwrite_addr[0]), .ADRA1(iwrite_addr[1]), .ADRA2(iwrite_addr[2]), 4.ADRA3(iwrite_addr[3]), // Write Address 5 .ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6(iwrite_addr[6]), // Write 6 Address 7 .DA0(ipixel_data5[0]), .DA1(ipixel_data5[1]), .DA2(ipixel_data5[2]), 8 .DA3(ipixel_data5[4]), .DA5(ipixel_data5[5]), .DA6(ipixel_data5[6]),	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1 .WEA(imem_wen), // Write enable 2 .MEA(vdd), // Memory enable 3 .ADRA0(iwrite_addr[0]), .ADRA1(iwrite_addr[1]), .ADRA2(iwrite_addr[2]), 4 .ADRA3(iwrite_addr[3]), // Write Address 5 .ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6(iwrite_addr[6]), // Write 6 .Address 7 .DA0(ipixel_data5[0]), .DA1(ipixel_data5[1]), .DA2(ipixel_data5[2]), 8 .DA3(ipixel_data5[3]), // Write Data 9 .DA4(ipixel_data5[4]), .DA5(ipixel_data5[5]), .DA6(ipixel_data5[6]), 10 .DA7(ipixel_data5[7]), // Write Data 11 .DA8(ipixel_data5[8]), .DA9(ipixel_data5[9]), .DA10(ipixel_data5[10]),	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1
Ex. 2113 - export_buffers.v 1	Ex. 2113 - export_buffers.v 1

1 2	.DA116(ipixel_data5[116]), .DA117(ipixel_data5[117]), .DA118(ipixel_data5[118]), .DA119(ipixel_data5[119]), // Write Data	1 .TDA36(ipixel_data5[36]), .TDA37(ipixel_data5[37]), .TDA38(ipixel_data5[38]), 2 .TDA39(ipixel_data5[39]), // Write Test Data
3 4	.DA120(ipixel_data5[120]), .DA121(ipixel_data5[121]), .DA122(ipixel_data5[122]), .DA123(ipixel_data5[123]), // Write Data	3 .TDA40(ipixel_data5[40]), .TDA41(ipixel_data5[41]), .TDA42(ipixel_data5[42]), 4 .TDA43(ipixel_data5[43]), // Write Test Data
5	DA124(ipixel_data5[124]),DA125(ipixel_data5[125]),DA126(ipixel_data5[126]),DA127(ipixel_data5[127]),DA127(ipixel_data5[127]),DA127(ipixel_data5[127]),DA127(ipixel_data5[127]),DA127(ipixel_data5[127]),DA127(ipixel_data5[127]),DA127(ipixel_data5[128]),DA127(ipixel_dat	5TDA44(ipixel_data5[44]),TDA45(ipixel_data5[45]),TDA46(ipixel_data5[46]), 6TDA47(ipixel_data5[47]), // Write Test Data
7	// WRITE TEST SIGNALS	7 .TDA48(ipixel_data5[48]), .TDA49(ipixel_data5[49]), .TDA50(ipixel_data5[50]),
8	.BISTEA(vss),	8 .TDA51(ipixel_data5[51]), // Write Test Data 9 .TDA52(ipixel_data5[52]), .TDA53(ipixel_data5[53]), .TDA54(ipixel_data5[54]),
9	.TWEA(vss), // Test write enable .TMEA(vss), // Test memory enable	10 .TDA55(ipixel_data5[55]), // Write Test Data 11 .TDA56(ipixel_data5[56]), .TDA57(ipixel_data5[57]), .TDA58(ipixel_data5[58]),
11 12	.TADRA0(iwrite_addr[0]), .TADRA1(iwrite_addr[1]), .TADRA2(iwrite_addr[2]), .TADRA3(iwrite_addr[3]), // Write Test Address	12 .TDA59(ipixel_data5[59]), // Write Test Data 13 .TDA60(ipixel_data5[60]), .TDA61(ipixel_data5[61]), .TDA62(ipixel_data5[62]),
13 14	.TADRA4(iwrite_addr[4]), .TADRA5(iwrite_addr[5]), .TADRA6(iwrite_addr[6]), // Write Test Address	14 .TDA63(ipixel_data5[63]), // Write Test Data
15 16	.TDA0(ipixel_data5[0]), .TDA1(ipixel_data5[1]), .TDA2(ipixel_data5[2]),	16 .TDA67(ipixel_data5[67]), // Write Test Data
17	.TDA3(ipixel_data5[3]), // Write Test Data .TDA4(ipixel_data5[4]),	17 .TDA68(ipixel_data5[68]), .TDA69(ipixel_data5[69]), .TDA70(ipixel_data5[70]), 18 .TDA71(ipixel_data5[71]), // Write Test Data
18 19	.TDA7(ipixel_data5[7]), // Write Test Data .TDA8(ipixel_data5[8]), .TDA9(ipixel_data5[9]), .TDA10(ipixel_data5[10]),	19 .TDA72(ipixel_data5[72]), .TDA73(ipixel_data5[73]), .TDA74(ipixel_data5[74]), 20 .TDA75(ipixel_data5[75]), // Write Test Data
20 21	.TDA11(ipixel_data5[11]), // Write Test Data .TDA12(ipixel_data5[12]), .TDA13(ipixel_data5[13]), .TDA14(ipixel_data5[14]),	21 .TDA76(ipixel_data5[76]), .TDA77(ipixel_data5[77]), .TDA78(ipixel_data5[78]), 22 .TDA79(ipixel_data5[79]), // Write Test Data
22	.TDA15(ipixel_data5[15]), // Write Test Data .TDA16(ipixel_data5[16]),TDA17(ipixel_data5[17]),TDA18(ipixel_data5[18]),	23 .TDA80(ipixel_data5[80]), .TDA81(ipixel_data5[81]), .TDA82(ipixel_data5[82]), 24 .TDA83(ipixel_data5[83]), // Write Test Data
24 25	.TDA20(ipixel data5[20]),	25 .TDA84(ipixel_data5[84]), .TDA85(ipixel_data5[85]), .TDA86(ipixel_data5[86]), 26 .TDA87(ipixel_data5[87]), // Write Test Data
26 27	. TDA24(pixel_data5[24]), TDA25(pixel_data5[25]), TDA26(pixel_data5[24]), TDA24(pixel_data5[24]), TDA25(pixel_data5[25]), TDA26(pixel_data5[26]),	27 .TDA88(ipixel_data5[88]), .TDA89(ipixel_data5[89]), .TDA90(ipixel_data5[90]), 28 .TDA91(ipixel_data5[91]), // Write Test Data
28	.TDA27(ipixel_data5[27]), // Write Test Data	29 .TDA92(ipixel_data5[92]), .TDA93(ipixel_data5[93]), .TDA94(ipixel_data5[94]), 30 .TDA95(ipixel_data5[95]), // Write Test Data
29 30	.TDA28(ipixel_data5[28]), .TDA29(ipixel_data5[29]), .TDA30(ipixel_data5[30]), .TDA31(ipixel_data5[31]), // Write Test Data	31 .TDA96(ipixel_data5[96]), .TDA97(ipixel_data5[97]), .TDA98(ipixel_data5[98]), 32 .TDA99(ipixel_data5[99]), // Write Test Data
31 32	.TDA32(ipixel_data5[32]), .TDA33(ipixel_data5[33]), .TDA34(ipixel_data5[34]), .TDA35(ipixel_data5[35]), // Write Test Data	
	Page 53 of 84 Ex. 2113 - export_buffers.v	Page 54 of 84 Ex. 2113 - export_buffers.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.TDA104(ipixel_data5[103]), .TDA101(ipixel_data5[101]), .TDA102(ipixel_data5[102]), .TDA103(ipixel_data5[103]),	1 dum_mem_p2 #(7,128) bank1_buff2(iRCLK(selk), 2 iWCLK(selk), 3 iMER(q1_mem_re), 4 iMEW(imem_wen), 5 iWEN(imem_wew), 6 iRADR(q1_read_addr[6:0]), 7 iWADR(iwrite_addr), 8 iD(ipixel_data6), 9 .oQ(buff6_out) 10); 11 'else // !'ifdef USE_BEHAVE_MEM 12 rfsd2_80x128cm2sw0 ubank1_buff2 13 (/*VRGIO_rfsd2_80x128cm2sw0_ipixel_data6_buff6_out_iwrite_addr_q1_read_addr_l4_imem_wen_q1_mem_re_null*/ 15 // READ INTERFACE
19 20	$. TADRB0(q0_read_addr[0]), . TADRB1(q0_read_addr[1]), . TADRB2(q0_read_addr[2]), \\ . TADRB3(q0_read_addr[3]), / Read Test Address$	16 .CLKB(iSCLK), // Read Clock 17 .OEB(vdd), // Output enable
21 22	$. TADRB4(q0_read_addr[4]), \ . TADRB5(q0_read_addr[5]), \ . TADRB6(q0_read_addr[6]), \ // \ Read\ Test\ Address$.MEB(q1_mem_re), // Read enable
23	.AWTB(vss)	19 .ADRB0(q1_read_addr[0]), .ADRB1(q1_read_addr[1]), .ADRB2(q1_read_addr[2]), 20 .ADRB3(q1_read_addr[3]), // Read Address
24 25);	21 .ADRB4(q1_read_addr[4]), .ADRB5(q1_read_addr[5]), .ADRB6(q1_read_addr[6]), // 22 Read Address
26	`endif // !`ifdef USE_BEHAVE_MEM	23 .QB0(buff6_out[0]), .QB1(buff6_out[1]), .QB2(buff6_out[2]), .QB3(buff6_out[3]), // Read 24 Data
27 28		25 .QB4(buff6_out[4]), .QB5(buff6_out[5]), .QB6(buff6_out[6]), .QB7(buff6_out[7]), // Read 26 Data
29	`ifdef USE_BEHAVE_MEM	27 .QB8(buff6_out[8]), .QB9(buff6_out[9]), .QB10(buff6_out[10]), .QB11(buff6_out[11]), // 28 Read Data
	Page 55 of 84	Page 56 of 84

1 2	.QB12(buff6_out[12]), .QB13(buff6_out[13]), .QB15(buff6_out[15]), // Read Data	.QB14(buff6_out[14]),	1 2	.QB76(buff6_out[76]), .QB77(buff6_out[77]), .QB79(buff6_out[79]), // Read Data	.QB78(buff6_out[78]),
3	.QB16(buff6_out[16]), .QB17(buff6_out[17]), .QB19(buff6_out[19]), // Read Data	.QB18(buff6_out[18]),	3 4	.QB80(buff6_out[80]), .QB81(buff6_out[81]), .QB83(buff6_out[83]), // Read Data	.QB82(buff6_out[82]),
5	.QB20(buff6_out[20]), .QB21(buff6_out[21]), .QB23(buff6_out[23]), // Read Data	.QB22(buff6_out[22]),	5 6	.QB84(buff6_out[84]), .QB85(buff6_out[85]), .QB87(buff6_out[87]), // Read Data	.QB86(buff6_out[86]),
7 8	.QB24(buff6_out[24]), .QB25(buff6_out[25]), .QB27(buff6_out[27]), // Read Data	.QB26(buff6_out[26]),	7 8	.QB88(buff6_out[88]), .QB89(buff6_out[89]), .QB91(buff6_out[91]), // Read Data	.QB90(buff6_out[90]),
9 10	.QB28(buff6_out[28]), .QB29(buff6_out[29]), .QB31(buff6_out[31]), // Read Data	.QB30(buff6_out[30]),	9 10	.QB92(buff6_out[92]), .QB93(buff6_out[93]), .QB95(buff6_out[95]), // Read Data	.QB94(buff6_out[94]),
11 12	.QB32(buff6_out[32]), .QB33(buff6_out[33]), .QB35(buff6_out[35]), // Read Data	.QB34(buff6_out[34]),	11 12	.QB96(buff6_out[96]), .QB97(buff6_out[97]), .QB99(buff6_out[99]), // Read Data	.QB98(buff6_out[98]),
13 14	.QB36(buff6_out[36]), .QB37(buff6_out[37]), .QB39(buff6_out[39]), // Read Data	.QB38(buff6_out[38]),	13 14	.QB100(buff6_out[100]), .QB101(buff6_out[101]), .QB103(buff6_out[103]), // Read Data	.QB102(buff6_out[102]),
15 16	.QB40(buff6_out[40]), .QB41(buff6_out[41]), .QB43(buff6_out[43]), // Read Data	.QB42(buff6_out[42]),	15 16	.QB104(buff6_out[104]), .QB105(buff6_out[105]), .QB107(buff6_out[107]), // Read Data	.QB106(buff6_out[106]),
17 18	.QB44(buff6_out[44]), .QB45(buff6_out[45]), .QB47(buff6_out[47]), // Read Data	.QB46(buff6_out[46]),	17 18	.QB108(buff6_out[108]), .QB109(buff6_out[109]), .QB111(buff6_out[111]), // Read Data	.QB110(buff6_out[110]),
19 20	.QB48(buff6_out[48]), .QB49(buff6_out[49]), .QB51(buff6_out[51]), // Read Data	.QB50(buff6_out[50]),	19 20	.QB112(buff6_out[112]), .QB113(buff6_out[113]), .QB115(buff6_out[115]), // Read Data	.QB114(buff6_out[114]),
21 22	.QB52(buff6_out[52]), .QB53(buff6_out[53]), .QB55(buff6_out[55]), // Read Data	.QB54(buff6_out[54]),	21 22	.QB116(buff6_out[116]), .QB117(buff6_out[117]), .QB119(buff6_out[119]), // Read Data	.QB118(buff6_out[118]),
23 24	.QB56(buff6_out[56]), .QB57(buff6_out[57]), .QB59(buff6_out[59]), // Read Data	.QB58(buff6_out[58]),	23 24	.QB120(buff6_out[120]), .QB121(buff6_out[121]), .QB123(buff6_out[123]), // Read Data	.QB122(buff6_out[122]),
25 26	.QB60(buff6_out[60]), .QB61(buff6_out[61]), .QB63(buff6_out[63]), // Read Data	.QB62(buff6_out[62]),	25 26	.QB124(buff6_out[124]), .QB125(buff6_out[125]), .QB127(buff6_out[127]), // Read Data	.QB126(buff6_out[126]),
27 28	.QB64(buff6_out[64]), .QB65(buff6_out[65]), .QB67(buff6_out[67]), // Read Data	.QB66(buff6_out[66]),	27	// WRITE INTERFACE	
29	.QB68(buff6_out[68]), .QB69(buff6_out[69]), .QB71(buff6_out[71]), // Read Data	.QB70(buff6_out[70]),	28 29	.CLKA(iSCLK), // Write Clock .WEA(imem_wen), // Write enable	
31 32	.QB72(buff6_out[72]), .QB73(buff6_out[73]), .QB75(buff6_out[75]), // Read Data	.QB74(buff6_out[74]),	30	.MEA(vdd), // Memory enable	
22	.xp/s(vano_van(13)), // read bala		31 32	.ADRA0(iwrite_addr[0]), .ADRA1(iwrite_addr[1]), .ADRA3(iwrite_addr[3]), // Write Address	.ADRA2(iwrite_addr[2]),
	Page 57 of 84	Ex. 2113 - export_buffers.v		Page 58 of 84	Ex. 2113 - export_buffers.v
1 2	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADR/		1 2	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]),	Ex. 2113 - export_buffers.v .DA62(ipixel_data6[62]),
1 2 3 4	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6ddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]),		1 2 3 4	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), // Write Data .DA64(ipixel_data6[64]), .DA65(ipixel_data6[65]),	
	ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADR/Address .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), // Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]),	.6(iwrite_addr[6]), // Write		.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), //Write Data .DA64(ipixel_data6[64]), .DA65(ipixel_data6[65]), .DA67(ipixel_data6[67]), //Write Data .DA68(ipixel_data6[68]), .DA69(ipixel_data6[68]), .DA69(ipixel_data6[68]), .DA69(ipixel_data6[68]), .DA69(ipixel_data6[69]),	.DA62(ipixel_data6[62]),
3 4 5	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6ddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[4]), .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA5(ipixel_data6[7]), .DA5(ipixel_data6[8]), .DA9(ipixel_data6[9]),	.6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]),	3 4 5	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), //Write Data .DA64(ipixel_data6[64]), .DA65(ipixel_data6[65]), .DA67(ipixel_data6[67]), //Write Data .DA68(ipixel_data6[68]), .DA69(ipixel_data6[69]), .DA71(ipixel_data6[71]), //Write Data .DA72(ipixel_data6[72]), .DA73(ipixel_data6[73]),	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]),
3 4 5 6	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADR/Address .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), // Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), // Write Data	.t6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]),	3 4 5 6	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), //Write Data .DA64(ipixel_data6[64]), .DA65(ipixel_data6[65]), .DA67(ipixel_data6[67]), //Write Data .DA68(ipixel_data6[68]), .DA69(ipixel_data6[69]), .DA71(ipixel_data6[71]), //Write Data	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]),
3 4 5 6 7 8	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6ddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), .// Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), .// Write Data .DA8(ipixel_data6[1]), .// Write Data .DA14(ipixel_data6[1]), .// Write Data .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]), .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]),	.6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]),	3 4 5 6 7 8	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), //Write Data .DA64(ipixel_data6[63]), .DA65(ipixel_data6[65]), .DA67(ipixel_data6[67]), //Write Data .DA68(ipixel_data6[68]), .DA69(ipixel_data6[69]), .DA71(ipixel_data6[71]), //Write Data .DA72(ipixel_data6[72]), .DA73(ipixel_data6[73]), .DA75(ipixel_data6[75]), //Write Data .DA76(ipixel_data6[75]), //Write Data	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]),
3 4 5 6 7 8 9 10	ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADR/Address DA0(ipixel_data6[0]),DA1(ipixel_data6[1]), .DA3(ipixel_data6[1]),DA5(ipixel_data6[4]),DA5(ipixel_data6[4]),DA5(ipixel_data6[4]),DA9(ipixel_data6[7]),DA9(ipixel_data6[9]), .DA11(ipixel_data6[1]),DA9(ipixel_data6[1]),DA13(ipixel_data6[12]),DA13(ipixel_data6[13]),DA15(ipixel_data6[14]),DA15(ipixel_data6[16]),DA17(ipixel_data6[17]),DA16(ipixel_data6[16]),DA17(ipixel_data6[17]),DA16(ipixel_data6[16]),DA17(ipixel_data6[17]),DA16(ipixel_data6[16]),DA17(ipixel_data6[17]),DA16(ipixel_data6[16]),DA17(ipixel_data6[17]),DA16(ipixel_data6[16]),DA17(ipixel_data6[17]),DA17(ipixel_data6[17]),DA17(ipixel_data6[17]),DA17(ipixel_data6[17]),DA17(ipixel_data6[17]),DA18(ipixel_data6[17]),DA17(ipixel_data6[17]),DA18(ipixe	.t6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]),	3 4 5 6 7 8 9 10	DA60(ipixel_data6[60]), DA61(ipixel_data6[61]), DA63(ipixel_data6[63]), //Write Data DA64(ipixel_data6[63]), //Write Data DA65(ipixel_data6[67]), //Write Data DA68(ipixel_data6[68]), DA69(ipixel_data6[69]), DA71(ipixel_data6[71]), //Write Data DA72(ipixel_data6[72]), DA73(ipixel_data6[73]), DA75(ipixel_data6[75]), //Write Data DA76(ipixel_data6[76]), DA77(ipixel_data6[77]), DA79(ipixel_data6[76]), //Write Data DA80(ipixel_data6[90]), DA81(ipixel_data6[81]),	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]),
3 4 5 6 7 8 9 10 11 12	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6ddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), // Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), // Write Data .DA8(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[1]), // Write Data .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]), .DA15(ipixel_data6[15]), // Write Data .DA16(ipixel_data6[16]), .DA17(ipixel_data6[17]), .DA19(ipixel_data6[19]), // Write Data .DA10(ipixel_data6[19]), // Write Data .DA10(ipixel_data6[19]), // Write Data	.b6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA18(ipixel_data6[18]),	3 4 5 6 7 8 9 10 11 12 13	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), //Write Data .DA64(ipixel_data6[63]), //Write Data .DA65(ipixel_data6[67]), //Write Data .DA68(ipixel_data6[68]), .DA69(ipixel_data6[69]), .DA71(ipixel_data6[71]), //Write Data .DA72(ipixel_data6[72]), .DA73(ipixel_data6[73]), .DA75(ipixel_data6[72]), .DA75(ipixel_data6[73]), //Write Data .DA76(ipixel_data6[73]), //Write Data .DA80(ipixel_data6[80]), .DA81(ipixel_data6[81]), .DA83(ipixel_data6[83]), //Write Data .DA84(ipixel_data6[84]), .DA85(ipixel_data6[85]),	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]),
3 4 5 6 7 8 9 10 11 12 13 14	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADR/Address .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), ./Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[5]), .DA9(ipixel_data6[5]), .DA9(ipixel_data6[6]), .DA9(ipixel_data6[6]), .DA11(ipixel_data6[1]), ./Write Data .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]), .DA15(ipixel_data6[13]), .DA15(ipixel_data6[13]), .DA17(ipixel_data6[13]), .DA19(ipixel_data6[20]), .DA21(ipixel_data6[21]), .DA23(ipixel_data6[23]), ./Write Data .DA24(ipixel_data6[24]),DA25(ipixel_data6[25]), .DA24(ipixel_data6[25]),DA25(ip	.6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA18(ipixel_data6[18]), .DA22(ipixel_data6[22]),	3 4 5 6 7 8 9 10 11 12 13 14	.DA60(ipixel_data6[60]), .DA61(ipixel_data6[61]), .DA63(ipixel_data6[63]), //Write Data .DA64(ipixel_data6[63]), //Write Data .DA67(ipixel_data6[67]), //Write Data .DA68(ipixel_data6[68]), .DA69(ipixel_data6[69]), .DA71(ipixel_data6[71]), //Write Data .DA72(ipixel_data6[72]), //Write Data .DA75(ipixel_data6[72]), //Write Data .DA76(ipixel_data6[76]), .DA77(ipixel_data6[73]), .DA79(ipixel_data6[79]), //Write Data .DA80(ipixel_data6[79]), //Write Data .DA83(ipixel_data6[83]), //Write Data .DA84(ipixel_data6[84]), .DA85(ipixel_data6[85]), .DA87(ipixel_data6[85]), .DA87(ipixel_data6[85]), .DA87(ipixel_data6[85]), .DA89(ipixel_data6[88]), .DA89(ipixel_data6[8	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA86(ipixel_data6[86]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6dress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[1]), .DA5(ipixel_data6[1]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA10(ipixel_data6[7]), .Write Data .DA3(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[11]), .Write Data .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]), .DA15(ipixel_data6[15]), .Write Data .DA16(ipixel_data6[16]), .DA17(ipixel_data6[17]), .DA19(ipixel_data6[21]), .DA20(ipixel_data6[23]), .Write Data .DA24(ipixel_data6[24]), .DA25(ipixel_data6[25]), .DA27(ipixel_data6[27]), .Write Data .DA24(ipixel_data6[27]), .Write Data .DA24(ipixel_data6[27]),DA25(ipixel_data6[25]), .DA27(ipixel_data6[27]),DA29(ipixel_data6[27]),	.t6(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA18(ipixel_data6[18]), .DA22(ipixel_data6[22]), .DA26(ipixel_data6[26]),	3 4 5 6 7 8 9 10 11 12 13 14 15 16	DA60(ipixel_data6[60]),DA61(ipixel_data6[61]),DA63(ipixel_data6[63]), // Write Data DA64(ipixel_data6[63]), // Write Data DA65(ipixel_data6[64]),DA65(ipixel_data6[65]),DA65(ipixel_data6[67]), // Write Data DA76(ipixel_data6[71]), // Write Data DA76(ipixel_data6[72]),DA73(ipixel_data6[73]),DA75(ipixel_data6[73]), // Write Data DA76(ipixel_data6[79]), // Write Data DA80(ipixel_data6[80]),DA81(ipixel_data6[81]),DA83(ipixel_data6[83]), // Write Data DA84(ipixel_data6[84]),DA85(ipixel_data6[85]),DA87(ipixel_data6[85]),DA89(ipixel_data6[85]),DA89(ipixel_data6[88]),DA89(ipixel_data6[87]), // Write Data DA88(ipixel_data6[91]), // Write Data DA91(ipixel_data6[92]),DA93(ipixel_data6[93]	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA86(ipixel_data6[82]), .DA90(ipixel_data6[90]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6ddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), .// Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), .// Write Data .DA8(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[1]), .// Write Data .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]), .DA15(ipixel_data6[15]), .// Write Data .DA16(ipixel_data6[16]), .DA17(ipixel_data6[17]), .DA19(ipixel_data6[20]), .DA21(ipixel_data6[21]), .DA23(ipixel_data6[23]), .// Write Data .DA24(ipixel_data6[24]), .DA25(ipixel_data6[25]), .DA27(ipixel_data6[25]), .DA28(ipixel_data6[28]), .DA29(ipixel_data6[28]), .DA29(ipixel_data6[28]), .DA29(ipixel_data6[31]), .// Write Data .DA28(ipixel_data6[31]), .// Write Data .DA32(ipixel_data6[31]), .// Write Data .DA32(ipixel_data6[31]), .// Write Data	DA2(ipixel_data6[2]), DA6(ipixel_data6[6]), DA10(ipixel_data6[10]), DA14(ipixel_data6[14]), DA18(ipixel_data6[18]), DA22(ipixel_data6[22]), DA26(ipixel_data6[26]), DA30(ipixel_data6[30]),	3 4 5 6 7 8 9 10 11 12 13 14 15 16	DA60(ipixel_data6[60]),DA61(ipixel_data6[61]),DA63(ipixel_data6[63]), // Write DataDA64(ipixel_data6[63]),DA65(ipixel_data6[65]),DA67(ipixel_data6[67]), // Write DataDA68(ipixel_data6[68]),DA69(ipixel_data6[68]),DA69(ipixel_data6[71]), // Write DataDA72(ipixel_data6[72]),DA73(ipixel_data6[73]),DA75(ipixel_data6[72]),DA75(ipixel_data6[72]),DA77(ipixel_data6[77]),DA79(ipixel_data6[79]), // Write DataDA80(ipixel_data6[80]),DA81(ipixel_data6[81]),DA81(ipixel_data6[83]),DA81(ipixel_data6[83]),DA84(ipixel_data6[83]),DA85(ipixel_data6[83]),DA85(ipixel_data6[83]),DA89(ipixel_data6[83]),DA91(ipixel_data6[83]),DA91(ipixel_data6[93]),DA91(ipixel_data6[93]),DA92(ipixel_data6[93]),DA93(ipixel_data6[93]),DA95(ipixel_data6[93]),DA95(ipixel_data6[93]),DA95(ipixel_data6[95]),DA97(ipixel_data	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA86(ipixel_data6[86]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[94]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	ADRA4(iwrite_addr[4]), ADRA5(iwrite_addr[5]), ADRA Address DA0(ipixel_data6[0]), DA1(ipixel_data6[1]), DA3(ipixel_data6[4]), DA5(ipixel_data6[1]), DA7(ipixel_data6[4]), DA9(ipixel_data6[5]), DA1(ipixel_data6[8]), DA9(ipixel_data6[9]), DA11(ipixel_data6[1]), Write Data DA12(ipixel_data6[12]), DA13(ipixel_data6[13]), DA15(ipixel_data6[15]), Write Data DA16(ipixel_data6[16]), DA17(ipixel_data6[17]), DA19(ipixel_data6[19]), Write Data DA20(ipixel_data6[23]), Write Data DA24(ipixel_data6[23]), "Write Data DA24(ipixel_data6[24]), DA25(ipixel_data6[25]), DA27(ipixel_data6[27]), Write Data DA28(ipixel_data6[27]), DA29(ipixel_data6[27]), DA31(ipixel_data6[32]), DA33(ipixel_data6[37]), DA32(ipixel_data6[32]), DA33(ipixel_data6[37]), DA33(ipixel_data6[35]), Write Data DA35(ipixel_data6[35]), "DA37(ipixel_data6[37]),	.06(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA22(ipixel_data6[18]), .DA26(ipixel_data6[22]), .DA30(ipixel_data6[30]), .DA30(ipixel_data6[30]),	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	DA60(ipixel_data6[60]),DA61(ipixel_data6[61]),DA63(ipixel_data6[63]),/Write DataDA64(ipixel_data6[63]),/Write DataDA65(ipixel_data6[64]),DA65(ipixel_data6[65]),DA69(ipixel_data6[65]),DA71(ipixel_data6[76]),DA73(ipixel_data6[72]),DA73(ipixel_data6[73]),DA73(ipixel_data6[73]),DA73(ipixel_data6[73]),DA73(ipixel_data6[73]),DA77(ipixel_data6[73]),DA77(ipixel_data6[73]),DA79(ipixel_data6[80]),DA81(ipixel_data6[81]),DA83(ipixel_data6[83]),DA81(ipixel_data6[83]),DA83(ipixel_data6[83]),DA85(ipixel_data6[83]),DA85(ipixel_data6[83]),DA89(ipixel_data6[91]),DA93(ipixel_data6[91]),DA93(ipixel_data6[92]),DA93(ipixel_data6[93]),DA93(ipixel_data6[93]),DA96(ipixel_data6[96]),DA97(ipixel_data6[97]),DA99(ipixel_data6[96]),DA99(ipixel_data6[97]),DA99(ipixel_data6[97]),DA99(ipixel_data6[97]),DA99(ipixel_data6[97]),DA99(ipixel_data6[96]),DA91(ipixel_data6[97]),DA99(ipixel_data6[97]),DA99(ipixel_data6[97]),DA91(ipixel_data6[97]),DA90(ipixel_data6[96]),DA91(ipixel_data6[97]),DA90(ipixel_data6[97]),DA91(ipixel_dat	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA86(ipixel_data6[82]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[94]), .DA98(ipixel_data6[98]),
3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6ddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[1]), .DA5(ipixel_data6[1]), .DA5(ipixel_data6[1]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), .Write Data .DA4(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[1]), .DA13(ipixel_data6[1]), .DA13(ipixel_data6[1]), .DA15(ipixel_data6[15]), .DA15(ipixel_data6[16]), .DA17(ipixel_data6[17]), .DA19(ipixel_data6[16]), .DA17(ipixel_data6[17]), .DA29(ipixel_data6[23]), .DA21(ipixel_data6[23]), .DA25(ipixel_data6[23]), .DA25(ipixel_data6[23]), .DA25(ipixel_data6[23]), .DA25(ipixel_data6[23]), .DA25(ipixel_data6[23]), .DA35(ipixel_data6[31]), .DA35(ipixel_data6[31]), .DA35(ipixel_data6[35]), .DA33(ipixel_data6[35]), .DA33(ipixel_data6[37]), .DA36(ipixel_data6[36]), .DA37(ipixel_data6[37]), .DA36(ipixel_data6[36]), .DA37(ipixel_data6[37]), .DA39(ipixel_data6[40]), .DA41(ipixel_data6[41]), .DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_data6[41]),DA41(ipixel_	.06(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA18(ipixel_data6[18]), .DA22(ipixel_data6[22]), .DA30(ipixel_data6[26]), .DA34(ipixel_data6[34]), .DA38(ipixel_data6[38]),	3 4 5 6 7 8 9 10 111 12 13 14 15 16 17 18 19 20 21 22 23	DA60(ipixel_data6[60]),DA61(ipixel_data6[61]),DA63(ipixel_data6[63]), // Write DataDA64(ipixel_data6[63]),DA65(ipixel_data6[65]),DA65(ipixel_data6[67]), // Write DataDA68(ipixel_data6[68]),DA69(ipixel_data6[69]),DA71(ipixel_data6[71]), // Write DataDA72(ipixel_data6[72]),DA73(ipixel_data6[73]),DA75(ipixel_data6[73]),DA75(ipixel_data6[73]), // Write DataDA80(ipixel_data6[79]), // Write DataDA80(ipixel_data6[80]),DA81(ipixel_data6[81]),DA89(ipixel_data6[84]),DA85(ipixel_data6[84]),DA85(ipixel_data6[84]),DA85(ipixel_data6[84]),DA89(ipixel_data6[87]), // Write DataDA88(ipixel_data6[87]), // Write DataDA92(ipixel_data6[97]), // Write DataDA92(ipixel_data6[97]),DA93(ipixel_data6[97]),DA95(ipixel_data6[97]),DA97(ipixel_data6[97]),DA97(ipixel_data6[97]),DA97(ipixel_data6[97]),DA97(ipixel_data6[97]),DA97(ipixel_data6[101]),DA97(ipixel_data6[101]),DA97(ipixel_data6[101]),DA104(ipixel_data6[101]),DA104(ipixel_data6[101]),DA104(ipixel_data6[105]),DA104(ipixel_data6[105]),DA104(ipixel_data6[105]),DA104(ipixel_data6[105]),DA105(ipixel_data6[105]),DA104(ipixel_data6[105]),DA105(ipixel_data6[105	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA86(ipixel_data6[82]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[94]), .DA98(ipixel_data6[94]), .DA98(ipixel_data6[102]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	ADRA4(iwrite_addr[4]), ADRA5(iwrite_addr[5]), ADR/Address DA0(ipixel_data6[0]), Write Data DA4(ipixel_data6[4]), DA5(ipixel_data6[1]), DA5(ipixel_data6[4]), DA5(ipixel_data6[4]), DA9(ipixel_data6[5]), DA1(ipixel_data6[7]), Write Data DA8(ipixel_data6[1]), Write Data DA12(ipixel_data6[12]), DA13(ipixel_data6[9]), DA11(ipixel_data6[12]), DA15(ipixel_data6[12]), Write Data DA16(ipixel_data6[16]), DA17(ipixel_data6[17]), DA19(ipixel_data6[19]), Write Data DA20(ipixel_data6[20]), DA21(ipixel_data6[21]), DA23(ipixel_data6[23]), Write Data DA24(ipixel_data6[24]), DA25(ipixel_data6[25]), DA31(ipixel_data6[27]), Write Data DA28(ipixel_data6[28]), DA29(ipixel_data6[29]), DA31(ipixel_data6[31]), Write Data DA32(ipixel_data6[32]), DA33(ipixel_data6[37]), DA36(ipixel_data6[37]), Write Data DA36(ipixel_data6[38]), DA37(ipixel_data6[37]), DA36(ipixel_data6[37]), DA37(ipixel_data6[37]), DA39(ipixel_data6[37]), DA39(ipixel_data6[40]), DA41(ipixel_data6[41]), DA44(ipixel_data6[44]), DA44(ipixel_data6[44]), DA44(ipixel_data6[44]), DA44(ipixel_data6[44]), DA44(ipixel_data6[44]), DA45(ipixel_data6[44]), DA44(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[44]), DA45(ipixel_data6[45]),	.06(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA18(ipixel_data6[14]), .DA22(ipixel_data6[18]), .DA22(ipixel_data6[22]), .DA30(ipixel_data6[26]), .DA30(ipixel_data6[34]), .DA38(ipixel_data6[34]), .DA42(ipixel_data6[42]),	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	DA60(ipixel_data6[60]), DA61(ipixel_data6[61]), DA63(ipixel_data6[63]), //Write Data DA64(ipixel_data6[63]), //Write Data DA64(ipixel_data6[63]), //Write Data DA68(ipixel_data6[71), //Write Data DA68(ipixel_data6[78]), //Write Data DA72(ipixel_data6[72]), DA73(ipixel_data6[69]), DA73(ipixel_data6[72]), //Write Data DA76(ipixel_data6[72]), //Write Data DA80(ipixel_data6[78]), //Write Data DA80(ipixel_data6[80]), DA81(ipixel_data6[81]), DA83(ipixel_data6[83]), //Write Data DA84(ipixel_data6[83]), //Write Data DA84(ipixel_data6[83]), //Write Data DA85(ipixel_data6[87]), //Write Data DA93(ipixel_data6[91]), //Write Data DA93(ipixel_data6[91]), //Write Data DA94(ipixel_data6[92]), DA93(ipixel_data6[93]), DA95(ipixel_data6[96]), //Write Data DA96(ipixel_data6[96]), //Write Data DA100(ipixel_data6[103]), //Write Data DA104(ipixel_data6[104]), //Write Data	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[90]), .DA94(ipixel_data6[94]), .DA98(ipixel_data6[98]), .DA102(ipixel_data6[102]), .DA106(ipixel_data6[106]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	ADRA4(iwrite_addr[4]), ADRA5(iwrite_addr[5]), ADRAddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[1]), .DA3(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), .Write Data .DA8(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[11]), .Write Data .DA12(ipixel_data6[12]), .DA13(ipixel_data6[13]), .DA15(ipixel_data6[15]), .Write Data .DA16(ipixel_data6[16]), .DA17(ipixel_data6[17]), .DA19(ipixel_data6[24]), .DA27(ipixel_data6[23]), .Write Data .DA24(ipixel_data6[23]), .Write Data .DA24(ipixel_data6[23]), .DA25(ipixel_data6[25]), .DA27(ipixel_data6[23]), .DA25(ipixel_data6[27]), .DA31(ipixel_data6[31]), .Write Data .DA32(ipixel_data6[32]), .DA33(ipixel_data6[33]), .DA35(ipixel_data6[35]), .DA37(ipixel_data6[37]), .DA36(ipixel_data6[36]), .DA37(ipixel_data6[37]), .DA39(ipixel_data6[36]), .DA37(ipixel_data6[41]), .DA46(ipixel_data6[44]), .DA47(ipixel_data6[44]), .DA45(ipixel_data6[45]), .DA47(ipixel_data6[45]), .Write Data .DA44(ipixel_data6[44]), .DA45(ipixel_data6[45]), .DA47(ipixel_data6[44]), .DA47(ipixel_data6[45]), .DA49(ipixel_data6[45]),DA49(ipixel_data6[45]),	.06(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA18(ipixel_data6[18]), .DA22(ipixel_data6[22]), .DA26(ipixel_data6[26]), .DA30(ipixel_data6[30]), .DA34(ipixel_data6[34]), .DA34(ipixel_data6[34]), .DA42(ipixel_data6[42]), .DA46(ipixel_data6[42]), .DA46(ipixel_data6[46]),	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27		.DA62(ipixel_data6[62]), .DA66(ipixel_data6[62]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA90(ipixel_data6[82]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[94]), .DA98(ipixel_data6[102]), .DA102(ipixel_data6[102]), .DA1106(ipixel_data6[100]), .DA110(ipixel_data6[110]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29	.ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6dress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), .DA3(ipixel_data6[3]), // Write Data .DA4(ipixel_data6[4]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), // Write Data .DA8(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[1]), // Write Data .DA12(ipixel_data6[1]), // Write Data .DA12(ipixel_data6[1]), // Write Data .DA16(ipixel_data6[1]), // Write Data .DA16(ipixel_data6[1]), // Write Data .DA20(ipixel_data6[2]), // Write Data .DA20(ipixel_data6[2]), // Write Data .DA24(ipixel_data6[2]), // Write Data .DA24(ipixel_data6[2]), // Write Data .DA28(ipixel_data6[2]), // Write Data .DA28(ipixel_data6[2]), // Write Data .DA28(ipixel_data6[3]), // Write Data .DA36(ipixel_data6[3]), // Write Data .DA36(ipixel_data6[3]), // Write Data .DA36(ipixel_data6[3]), // Write Data .DA36(ipixel_data6[3]), // Write Data .DA44(ipixel_data6[4]), .DA41(ipixel_data6[4]), .DA43(ipixel_data6[4]), .DA41(ipixel_data6[4]), .DA44(ipixel_data6[4]), .DA45(ipixel_data6[4]), // Write Data .DA48(ipixel_data6[4]), // Write Data	.06(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA18(ipixel_data6[14]), .DA22(ipixel_data6[18]), .DA22(ipixel_data6[22]), .DA30(ipixel_data6[26]), .DA30(ipixel_data6[30]), .DA38(ipixel_data6[34]), .DA42(ipixel_data6[41]), .DA46(ipixel_data6[42]), .DA46(ipixel_data6[46]), .DA50(ipixel_data6[40]),	3 4 5 6 7 8 9 10 111 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29	DA60(ipixel_data6[60]),DA61(ipixel_data6[61]),DA63(ipixel_data6[63]), // Write DataDA64(ipixel_data6[63]),DA65(ipixel_data6[65]),DA65(ipixel_data6[65]),DA65(ipixel_data6[65]),DA65(ipixel_data6[67]), // Write DataDA72(ipixel_data6[72]),DA69(ipixel_data6[73]),DA73(ipixel_data6[72]),DA73(ipixel_data6[73]),DA73(ipixel_data6[72]),DA73(ipixel_data6[73]),DA75(ipixel_data6[72]),DA77(ipixel_data6[73]),DA75(ipixel_data6[72]),DA77(ipixel_data6[72]),DA75(ipixel_data6[72]),DA81(ipixel_data6[81]),DA80(ipixel_data6[80]),DA81(ipixel_data6[81]),DA83(ipixel_data6[84]),DA85(ipixel_data6[85]),DA87(ipixel_data6[87]), // Write DataDA84(ipixel_data6[87]),DA95(ipixel_data6[91]),DA99(ipixel_data6[91]),DA93(ipixel_data6[92]),DA93(ipixel_data6[92]),DA93(ipixel_data6[93]),DA96(ipixel_data6[93]),DA97(ipixel_data6[93]),DA97(ipixel_data6[93]),DA97(ipixel_data6[103]),DA104(ipixel_data6[103]),DA104(ipixel_data6[103]),DA105(ipixel_data6[103]),DA105(ipixel_data6[103]),DA108(ipixel_data6[103]),DA109(ipixel_data6[103]),DA109(ipixel_data6[103]),DA109(ipixel_data6[103]),DA109(ipixel_data6[103]),DA109(ipixel_data6[103]),DA109(ipixel_data6[113]),DA109(ipixel_data6[113]),DA109(ipixel_data6[113]),DA116(ipixe	.DA62(ipixel_data6[62]), .DA66(ipixel_data6[66]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA90(ipixel_data6[86]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[94]), .DA98(ipixel_data6[98]), .DA102(ipixel_data6[102]), .DA106(ipixel_data6[106]), .DA110(ipixel_data6[110]), .DA114(ipixel_data6[111]),
3 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	ADRA4(iwrite_addr[4]), ADRA5(iwrite_addr[5]), ADRAddress .DA0(ipixel_data6[0]), .DA1(ipixel_data6[1]), DA3(ipixel_data6[1]), .DA5(ipixel_data6[1]), .DA5(ipixel_data6[1]), .DA5(ipixel_data6[5]), .DA7(ipixel_data6[7]), .Write Data .DA8(ipixel_data6[8]), .DA9(ipixel_data6[9]), .DA11(ipixel_data6[1]), .DA9(ipixel_data6[1]), .DA13(ipixel_data6[1]), .DA13(ipixel_data6[1]), .DA12(ipixel_data6[1]), .DA13(ipixel_data6[1]), .DA15(ipixel_data6[1]), .DA19(ipixel_data6[1]), .DA19(ipixel_data6[1]), .DA21(ipixel_data6[1]), .DA29(ipixel_data6[23]), .Write Data .DA24(ipixel_data6[23]), .DA25(ipixel_data6[25]), .DA27(ipixel_data6[23]), .DA29(ipixel_data6[27]), .DA31(ipixel_data6[27]), .DA31(ipixel_data6[31]), .DA39(ipixel_data6[31]), .DA39(ipixel_data6[31]), .DA39(ipixel_data6[31]), .DA39(ipixel_data6[34]), .DA31(ipixel_data6[41]), .DA49(ipixel_data6[41]), .DA41(ipixel_data6[41]), .DA41(ipixel_data6[42]), .DA41(ipixel_data6[42]), .DA41(ipixel_data6[43]),DA41(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixel_data6[43]),DA43(ipixe	.06(iwrite_addr[6]), // Write .DA2(ipixel_data6[2]), .DA6(ipixel_data6[6]), .DA10(ipixel_data6[10]), .DA14(ipixel_data6[14]), .DA18(ipixel_data6[18]), .DA22(ipixel_data6[22]), .DA26(ipixel_data6[26]), .DA30(ipixel_data6[30]), .DA34(ipixel_data6[34]), .DA38(ipixel_data6[34]), .DA42(ipixel_data6[42]), .DA40(ipixel_data6[42]), .DA50(ipixel_data6[50]), .DA50(ipixel_data6[50]), .DA54(ipixel_data6[50]),	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31		.DA62(ipixel_data6[62]), .DA66(ipixel_data6[62]), .DA70(ipixel_data6[70]), .DA74(ipixel_data6[74]), .DA78(ipixel_data6[78]), .DA82(ipixel_data6[82]), .DA86(ipixel_data6[82]), .DA90(ipixel_data6[90]), .DA94(ipixel_data6[94]), .DA98(ipixel_data6[94]), .DA102(ipixel_data6[102]), .DA106(ipixel_data6[102]), .DA110(ipixel_data6[110]), .DA114(ipixel_data6[111]), .DA118(ipixel_data6[114]),

1 2 2 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 14 15 16 16 17 18 12 22 23 3 24 25 26 27 28 29 30 31 32	DA124(ipixel_data6[124]),DA125(ipixel_data6[125]),DA126(ipixel_data6[126]),DA127(ipixel_data6[127]), // Write Data // WRITE TEST SIGNALS BISTEA(vss), // Test write enableTMEA(vss), // Test memory enableTADRA0(iwrite_addr[0]),TADRA1(iwrite_addr[1]),TADRA2(iwrite_addr[2]),TADRA3(iwrite_addr[3]), // Write Test AddressTADRA4(iwrite_addr[4]),TADRA5(iwrite_addr[5]),TADRA6(iwrite_addr[6]), // Write Test AddressTDA0(ipixel_data6[3]), // Write Test DataTDA4(ipixel_data6[3]), // Write Test DataTDA4(ipixel_data6[3]), // Write Test DataTDA4(ipixel_data6[4]),TDA5(ipixel_data6[5]),TDA6(ipixel_data6[6]),TDA1(ipixel_data6[1]),TDA1(ipixel_data6[1]),TDA1(ipixel_data6[1]),TDA1(ipixel_data6[1]),TDA1(ipixel_data6[1]),TDA15(ipixel_data6[1]),TDA15(ipixel_data6[1]),TDA15(ipixel_data6[1]),TDA16(ipixel_data6[1]),TDA16(ipixel_data6[1]),TDA16(ipixel_data6[1]),TDA16(ipixel_data6[2]),TDA17(ipixel_data6[2]),TDA20(ipixel_data6[2]),TDA21(ipixel_data6[2]),TDA22(ipixel_data6[2]),TDA23(ipixel_data6[2]),TDA23(ipixel_data6[2]),TDA24(ipixel_data6[2]),TDA25(ipixel_data6[2]),TDA25(ipixel_data6[2]),TDA36(ipixel_data6[2]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA37(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA37(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(ipixel_data6[3]),TDA36(1TDA44(ipixel_data6[44]),TDA45(ipixel_data6[45]),TDA46(ipixel_data6[46]), 2TDA47(ipixel_data6[47]), // Write Test Data 3TDA48(ipixel_data6[48]),TDA49(ipixel_data6[49]), 4TDA51(ipixel_data6[52]),TDA53(ipixel_data6[52]),TDA53(ipixel_data6[52]),TDA53(ipixel_data6[55]),TDA53(ipixel_data6[55]),TDA55(ipixel_data6[55]),TDA55(ipixel_data6[55]),TDA57(ipixel_data6[54]),TDA56(ipixel_data6[55]),TDA56(ipixel_data6[58]),TDA56(ipixel_data6[58]),TDA56(ipixel_data6[60]),TDA56(ipixel_data6[60]),TDA65(ipixel_data6[60]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[63]),TDA65(ipixel_data6[71]),TDA75(ipixel_data6[71]),TDA75(ipixel_data6[71]),TDA75(ipixel_data6[71]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[73]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[83]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[94]),TDA75(ipixel_data6[95]),TDA75(ipixel_data6[95]),TDA75(ipixel_data6[95]),TDA75(ipixel_data6[95]),
1 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28	.TDA108(ipixel_data6[108]), .TDA109(ipixel_data6[109]), .TDA110(ipixel_data6[110]), .TDA111(ipixel_data6[111]), // Write Test Data	1 .iWEN(imem_wew), 2 .iRADR(q2_read_addr[6:0]), 3 .iWADR(iwrite_addr), 4 .iD(ipixel_data7), 5 .oQ(buft7_out) 6 .); 7 'else // l'ifdef USE_BEHAVE_MEM 8 rfsd2_80x128cm2sw0 ubank1_buff3 9 (/*VRGIO_rfsd2_80x128cm2sw0 ipixel_data7_buff7_out iwrite_addr_q2_read_addr imem_wen_q2_mem_re_null*/ 11 // READ_INTERFACE 12 .CLKB(iSCLK), // Read_Clock 13 .OEB(vdd), // Output enable 14 .MEB(q2_mem_re), // Read_enable 15 .ADRB0(q2_read_addr[0]), .ADRB1(q2_read_addr[1]), .ADRB2(q2_read_addr[2]), 16 .ADRB3(q2_read_addr[4]), ./ Read_Address 17 .ADRB4(q2_read_addr[4]), .ADRB5(q2_read_addr[5]), .ADRB6(q2_read_addr[6]), // 8 Read_Address 19 .QB0(buff7_out[0]), .QB1(buff7_out[1]), .QB2(buff7_out[6]), .QB7(buff7_out[7]), // Read 20 .Data 21 .QB4(buff7_out[4]), .QB5(buff7_out[9]), .QB6(buff7_out[6]), .QB7(buff7_out[7]), // Read 22 .QB8(buff7_out[8]), .QB9(buff7_out[9]), .QB10(buff7_out[10]), .QB14(buff7_out[14]), 24 .QB15(buff7_out[12]),
	Page 63 of 84 Ex. 2113 - export_buffers.v	Page 64 of 84 Ex. 2113 - export_buffers.v

1	.QB20(buff7_out[20]), .QB21(buff7_out[21]),	.QB22(buff7_out[22]),	1 QB84(buff7_out[84]), QB85(buff7_out[85]), QB86(buff7_out[86]),
3 4	.QB23(buff7_out[23]), // Read Data .QB24(buff7_out[24]), .QB25(buff7_out[25]), .QB27(buff7_out[27]), // Read Data	.QB26(buff7_out[26]),	2 .QB87(buff7_out[87]), // Read Data 3 .QB88(buff7_out[88]), .QB89(buff7_out[89]), .QB90(buff7_out[90]), 4 .QB91(buff7_out[91]), // Read Data
5	.QB28(buff7_out[28]), .QB29(buff7_out[29]), .QB31(buff7_out[31]), // Read Data	.QB30(buff7_out[30]),	5 .QB92(buff7_out[92]), .QB93(buff7_out[93]), .QB94(buff7_out[94]), 6 .QB95(buff7_out[95]), // Read Data
7 8	.QB32(buff7_out[32]), .QB33(buff7_out[33]), .QB35(buff7_out[35]), // Read Data	.QB34(buff7_out[34]),	7 .QB96(buff7_out[96]), .QB97(buff7_out[97]), .QB98(buff7_out[98]), 8 .QB99(buff7_out[99]), // Read Data
9 10	.QB36(buff7_out[36]), .QB37(buff7_out[37]), .QB39(buff7_out[39]), // Read Data	.QB38(buff7_out[38]),	9
11 12	.QB40(buff7_out[40]), .QB41(buff7_out[41]), .QB43(buff7_out[43]), // Read Data	.QB42(buff7_out[42]),	11
13 14	.QB44(buff7_out[44]), .QB45(buff7_out[45]), .QB47(buff7_out[47]), // Read Data	.QB46(buff7_out[46]),	13 .QB108(buff7_out[108]), .QB109(buff7_out[109]), .QB110(buff7_out[110]), 14 .QB111(buff7_out[111]), // Read Data
15 16	.QB48(buff7_out[48]), .QB51(buff7_out[51]), // Read Data .QB51(buff7_out[51]), // Read Data	.QB50(buff7_out[50]),	15 .QB112(buff7_out[112]), .QB113(buff7_out[113]), .QB114(buff7_out[114]), .QB115(buff7_out[115]), // Read Data
17 18	.QB52(buff7_out[52]), .QB53(buff7_out[53]), .QB55(buff7_out[55]), $/\!/$ Read Data	.QB54(buff7_out[54]),	17 .QB116(buff7_out[116]), .QB117(buff7_out[117]), .QB118(buff7_out[118]), 18 .QB119(buff7_out[119]), // Read Data
19 20	$. QB56(buff7_out[56]), \\ . QB59(buff7_out[59]), \ // \ Read \ Data$.QB58(buff7_out[58]),	19 .QB120(bufi7_out[120]), .QB121(bufi7_out[121]), .QB122(bufi7_out[122]), 20 .QB123(bufi7_out[123]), // Read Data
21 22	$. QB60(buff7_out[60]), \\ . QB63(buff7_out[63]), \ // \ Read \ Data$ $. QB61(buff7_out[61]), \\ . QB63(buff7_out[63]), \ // \ Read \ Data$.QB62(buff7_out[62]),	21 .QB124(buft7_out[124]), .QB125(buft7_out[125]), .QB126(buff7_out[126]), 22 .QB127(buft7_out[127]), // Read Data
23 24	$.QB64(buff7_out[64]), \\ .QB67(buff7_out[67]), \ // \ Read \ Data$ $.QB65(buff7_out[65]), \\ .QB67(buff7_out[67]), \ // \ Read \ Data$.QB66(buff7_out[66]),	23 // WRITE INTERFACE 24 .CLKA(iSCLK), // Write Clock
25 26	.QB68(buff7_out[68]), .QB69(buff7_out[69]), .QB71(buff7_out[71]), // Read Data	.QB70(buff7_out[70]),	25 .WEA(imem_wen), // Write enable
27 28	.QB72(buff7_out[72]), .QB73(buff7_out[73]), .QB75(buff7_out[75]), // Read Data	.QB74(buff7_out[74]),	26 .MEA(vdd), // Memory enable
29 30	.QB75(buff7_out[75]), // Read Data .QB76(buff7_out[76]), .QB77(buff7_out[77]), .QB79(buff7_out[79]), // Read Data	.QB78(buff7_out[78]),	27 .ADRA0(iwrite_addr[0]), .ADRA1(iwrite_addr[1]), .ADRA2(iwrite_addr[2]), 28 .ADRA3(iwrite_addr[3]), // Write Address
31 32	.QB80(buff7_out[80]), .QB81(buff7_out[81]), .QB83(buff7_out[83]), // Read Data	.QB82(buff7_out[82]),	29 .ADRA4(iwrite_addr[4]), .ADRA5(iwrite_addr[5]), .ADRA6(iwrite_addr[6]), // Write 30 Address
			31 .DA0(ipixel_data7[0]), .DA1(ipixel_data7[1]), .DA2(ipixel_data7[2]), 32 .DA3(ipixel_data7[3]), // Write Data
	Page 65 of 84	Ex. 2113 - export_buffers.v	Page 66 of 84 Ex. 2113 - export_buffers.v
1 2	.DA4(ipixel_data7[4]), .DA5(ipixel_data7[5]), .DA7(ipixel_data7[7]), // Write Data	.DA6(ipixel_data7[6]),	1 .DA68(ipixel_data7[68]), .DA69(ipixel_data7[69]), .DA70(ipixel_data7[70]), 2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]),
3 4	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]), .DA9(ipixel_data7[9]), .DA11(ipixel_data7[11]), // Write Data	.DA10(ipixel_data7[10]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), 4 .DA75(ipixel_data7[75]), // Write Data
3	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]), .DA9(ipixel_data7[9]), .DA11(ipixel_data7[11]), // Write Data .DA12(ipixel_data7[12]), .DA13(ipixel_data7[13]), .DA15(ipixel_data7[15]), // Write Data	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), 4 .DA75(ipixel_data7[75]), // Write Data 5 .DA76(ipixel_data7[76]), .DA77(ipixel_data7[77]), .DA78(ipixel_data7[78]), 6 .DA79(ipixel_data7[79]), // Write Data
3 4 5 6 7 8	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]), DA9(ipixel_data7[9]), .DA11(ipixel_data7[11]), // Write Data .DA12(ipixel_data7[12]), DA13(ipixel_data7[13]), .DA15(ipixel_data7[15]), // Write Data .DA16(ipixel_data7[16]), DA17(ipixel_data7[17]), .DA19(ipixel_data7[19]), // Write Data	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), 4 .DA75(ipixel_data7[75]), // Write Data 5 .DA76(ipixel_data7[76]), .DA77(ipixel_data7[77]), .DA78(ipixel_data7[78]), 6 .DA79(ipixel_data7[79]), // Write Data 7 .DA80(ipixel_data7[80]), .DA81(ipixel_data7[81]), .DA82(ipixel_data7[82]), 8 .DA83(ipixel_data7[83]), // Write Data
3 4 5 6 7 8 9	.DA7(ipixel_data7[7]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), 4 .DA75(ipixel_data7[75]), // Write Data 5 .DA76(ipixel_data7[76]), .DA77(ipixel_data7[77]), .DA78(ipixel_data7[78]), 6 .DA79(ipixel_data7[79]), // Write Data 7 .DA80(ipixel_data7[80]), .DA81(ipixel_data7[81]), .DA82(ipixel_data7[82]), 8 .DA83(ipixel_data7[83]), // Write Data 9 .DA84(ipixel_data7[84]), .DA85(ipixel_data7[85]), .DA86(ipixel_data7[86]), 10 .DA87(ipixel_data7[87]), // Write Data
3 4 5 6 7 8 9 10	.DA7(ipixel_data7[7]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), DA74(ipixel_data7[74]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), DA78(ipixel_data7[78]), 6 DA90(ipixel_data7[79]), // Write Data 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), DA82(ipixel_data7[82]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), DA86(ipixel_data7[86]), 10 DA87(ipixel_data7[87]), // Write Data 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), DA90(ipixel_data7[90]), 12 DA91(ipixel_data7[91]), // Write Data
3 4 5 6 7 8 9 10	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), DA74(ipixel_data7[74]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), DA78(ipixel_data7[78]), 6 DA97(ipixel_data7[79]), // Write Data 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), DA82(ipixel_data7[82]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), DA86(ipixel_data7[86]), 10 DA87(ipixel_data7[71]), // Write Data 11 DA88(ipixel_data7[91]), // Write Data 12 DA99(ipixel_data7[91]), // Write Data 13 DA92(ipixel_data7[92]), DA93(ipixel_data7[93]), 14 DA95(ipixel_data7[95]), // Write Data 15 DA96(ipixel_data7[95]), // Write Data 16 DA96(ipixel_data7[95]), // Write Data
3 4 5 6 7 8 9 10 11 12 13 14	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]), DA9(ipixel_data7[9]), .DA11(ipixel_data7[11]), // Write Data .DA12(ipixel_data7[12]), DA13(ipixel_data7[13]), .DA15(ipixel_data7[12]), DA13(ipixel_data7[13]), .DA16(ipixel_data7[16]), DA17(ipixel_data7[17]), .DA19(ipixel_data7[19]), // Write Data .DA20(ipixel_data7[20]), DA21(ipixel_data7[21]), .DA23(ipixel_data7[23]), // Write Data .DA24(ipixel_data7[24]), DA25(ipixel_data7[25]), .DA27(ipixel_data7[28]), DA29(ipixel_data7[29]), .DA31(ipixel_data7[32]), // Write Data .DA32(ipixel_data7[32]), DA33(ipixel_data7[33]), .DA35(ipixel_data7[35]), // Write Data .DA36(ipixel_data7[36]), DA37(ipixel_data7[37]), DA36(ipixel_data7[36]), DA37(ipixel_data7[37]), DA36(ipixel_data7[36]), DA37(ipixel_data7[37]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[22]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), 4 .DA75(ipixel_data7[75]), // Write Data 5 .DA76(ipixel_data7[76]), .DA77(ipixel_data7[77]), .DA78(ipixel_data7[78]), 6 .DA79(ipixel_data7[78]), // Write Data 7 .DA80(ipixel_data7[80]), .DA81(ipixel_data7[81]), .DA82(ipixel_data7[82]), 8 .DA83(ipixel_data7[83]), // Write Data 9 .DA84(ipixel_data7[84]), .DA85(ipixel_data7[85]), .DA86(ipixel_data7[86]), 10 .DA87(ipixel_data7[87]), // Write Data 11 .DA88(ipixel_data7[88]), .DA89(ipixel_data7[89]), .DA90(ipixel_data7[90]), 12 .DA91(ipixel_data7[91]), // Write Data 13 .DA92(ipixel_data7[92]), .DA93(ipixel_data7[93]), .DA94(ipixel_data7[94]), 14 .DA95(ipixel_data7[95]), // Write Data 15 .DA96(ipixel_data7[96]), .DA97(ipixel_data7[97]), .DA98(ipixel_data7[98]), 16 .DA99(ipixel_data7[100]), .DA9101(ipixel_data7[101]), .DA9102(ipixel_data7[102]),
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), 4 .DA75(ipixel_data7[75]), // Write Data 5 .DA76(ipixel_data7[76]), .DA77(ipixel_data7[77]), .DA78(ipixel_data7[78]), 6 .DA79(ipixel_data7[78]), // Write Data 7 .DA80(ipixel_data7[80]), .DA81(ipixel_data7[81]), .DA82(ipixel_data7[82]), 8 .DA83(ipixel_data7[83]), // Write Data 9 .DA84(ipixel_data7[84]), .DA85(ipixel_data7[85]), .DA86(ipixel_data7[86]), 10 .DA87(ipixel_data7[88]), // Write Data 11 .DA88(ipixel_data7[88]), .DA89(ipixel_data7[89]), 12 .DA91(ipixel_data7[91]), // Write Data 13 .DA92(ipixel_data7[92]), .DA93(ipixel_data7[93]), .DA94(ipixel_data7[94]), 14 .DA95(ipixel_data7[95]), // Write Data 15 .DA96(ipixel_data7[96]), .DA97(ipixel_data7[97]), .DA98(ipixel_data7[98]), 16 .DA99(ipixel_data7[106]), .DA97(ipixel_data7[101]), .DA102(ipixel_data7[102]), 18 .DA103(ipixel_data7[103]), // Write Data 19 .DA104(ipixel_data7[103]), // Write Data 19 .DA104(ipixel_data7[104]), .DA105(ipixel_data7[105]), .DA106(ipixel_data7[106]),
3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[8]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]),	2 .DA71(ipixel_data7[71]), // Write Data 3 .DA72(ipixel_data7[72]), .DA73(ipixel_data7[73]), .DA74(ipixel_data7[74]), .DA75(ipixel_data7[75]), // Write Data 5 .DA76(ipixel_data7[76]), .DA77(ipixel_data7[77]), .DA78(ipixel_data7[78]), .DA78(ipixel_data7[78]), .DA78(ipixel_data7[78]), .DA80(ipixel_data7[80]), .DA81(ipixel_data7[81]), .DA82(ipixel_data7[82]), .DA83(ipixel_data7[83]), .// Write Data 9 .DA84(ipixel_data7[84]), .DA85(ipixel_data7[85]), .DA86(ipixel_data7[86]), .DA87(ipixel_data7[87]), // Write Data 11 .DA88(ipixel_data7[88]), .DA89(ipixel_data7[89]), .DA90(ipixel_data7[90]), .DA91(ipixel_data7[91]), // Write Data 13 .DA92(ipixel_data7[92]), .DA93(ipixel_data7[93]), .DA94(ipixel_data7[94]), .DA95(ipixel_data7[95]), // Write Data 15 .DA96(ipixel_data7[96]), .DA97(ipixel_data7[97]), .DA98(ipixel_data7[98]), .DA99(ipixel_data7[103]), // Write Data 17 .DA104(ipixel_data7[100]), .DA101(ipixel_data7[101]), .DA102(ipixel_data7[102]), .DA104(ipixel_data7[104]), .DA105(ipixel_data7[104]), .DA105(ipixel_data7[106]), .DA106(ipixel_data7[106]), .DA106(ipixel_data7[107]), // Write Data 21 .DA108(ipixel_data7[108]), .DA109(ipixel_data7[109]), .DA110(ipixel_data7[110]), .DA109(ipixel_data7[109]), .DA110(ipixel_data7[110]), .DA109(ipixel_data7[109]), .DA110(ipixel_data7[110]), .DA110(ipixel_data7[110])
3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 23	.DA7(ipixel_data7[7]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]), .DA42(ipixel_data7[42]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), DA74(ipixel_data7[74]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), DA78(ipixel_data7[78]), 6 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), DA78(ipixel_data7[78]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), DA82(ipixel_data7[82]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), DA86(ipixel_data7[86]), 10 DA87(ipixel_data7[87]), // Write Data 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), 12 DA91(ipixel_data7[91]), // Write Data 13 DA92(ipixel_data7[91]), // Write Data 14 DA95(ipixel_data7[92]), DA93(ipixel_data7[93]), DA94(ipixel_data7[94]), 15 DA96(ipixel_data7[99]), // Write Data 15 DA96(ipixel_data7[99]), // Write Data 16 DA96(ipixel_data7[104]), DA97(ipixel_data7[101]), DA106(ipixel_data7[102]), 17 DA106(ipixel_data7[104]), DA105(ipixel_data7[105]), DA106(ipixel_data7[106]), DA107(ipixel_data7[108]), DA106(ipixel_data7[106]), DA107(ipixel_data7[108]), DA106(ipixel_data7[110]), DA110(ipixel_data7[111]), DA1114(ipixel_data7[111]), DA114(ipixel_data7[111]), DA114(ipixel_data7[111]), DA114(ipixel_data7[114]), DA114(
3 4 4 5 6 6 7 8 8 9 9 10 11 12 13 14 15 16 17 18 19 20 12 22 23 24 25 5	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[1]), // Write Data .DA8(ipixel_data7[1]), // Write Data .DA12(ipixel_data7[12]), .DA13(ipixel_data7[13]), .DA15(ipixel_data7[12]), .DA13(ipixel_data7[13]), .DA15(ipixel_data7[12]), // Write Data .DA16(ipixel_data7[12]), // Write Data .DA20(ipixel_data7[20]), .DA21(ipixel_data7[21]), .DA19(ipixel_data7[24]), .DA25(ipixel_data7[24]), .DA25(ipixel_data7[25]), .DA27(ipixel_data7[24]), .DA25(ipixel_data7[27]), // Write Data .DA28(ipixel_data7[28]), .DA29(ipixel_data7[29]), .DA31(ipixel_data7[31]), // Write Data .DA32(ipixel_data7[32]), .DA33(ipixel_data7[33]), .DA35(ipixel_data7[35]), // Write Data .DA36(ipixel_data7[35]), // Write Data .DA36(ipixel_data7[36]), .DA37(ipixel_data7[47]), .DA39(ipixel_data7[44]), .DA41(ipixel_data7[44]), .DA41(ipixel_data7[45]), .DA44(ipixel_data7[44]), .DA45(ipixel_data7[45]), .DA48(ipixel_data7[44]), .DA49(ipixel_data7[45]), .DA49(ipixel_data7[47]), // Write Data .DA48(ipixel_data7[48]), .DA49(ipixel_data7[49]), .DA48(ipixel_data7[47]), // Write Data .DA48(ipixel_data7[48]), .DA49(ipixel_data7[49]), .DA51(ipixel_data7[45]), // Write Data .DA52(ipixel_data7[52]), .DA53(ipixel_data7[53]),	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]), .DA42(ipixel_data7[42]), .DA46(ipixel_data7[46]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 6 DA79(ipixel_data7[76]), DA77(ipixel_data7[77]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), 10 DA88(ipixel_data7[84]), DA85(ipixel_data7[85]), 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), 12 DA91(ipixel_data7[71]), // Write Data 13 DA92(ipixel_data7[71]), // Write Data 14 DA95(ipixel_data7[79]), DA93(ipixel_data7[93]), 15 DA96(ipixel_data7[96]), DA97(ipixel_data7[97]), 16 DA99(ipixel_data7[99]), // Write Data 17 DA100(ipixel_data7[100]), DA101(ipixel_data7[101]), DA102(ipixel_data7[102]), 18 DA104(ipixel_data7[104]), DA105(ipixel_data7[105]), DA106(ipixel_data7[106]), 20 DA107(ipixel_data7[107]), // Write Data 21 DA108(ipixel_data7[107]), // Write Data 22 DA111(ipixel_data7[11]), // Write Data 23 DA112(ipixel_data7[11]), // Write Data 24 DA115(ipixel_data7[11]), // Write Data 25 DA116(ipixel_data7[115]), // Write Data 26 DA116(ipixel_data7[115]), // Write Data 27 DA116(ipixel_data7[115]), // Write Data 28 DA116(ipixel_data7[115]), // Write Data 29 DA116(ipixel_data7[115]), // Write Data 20 DA116(ipixel_data7[115]), // Write Data 21 DA116(ipixel_data7[115]), // Write Data 22 DA116(ipixel_data7[115]), // Write Data 23 DA116(ipixel_data7[115]), // Write Data 24 DA116(ipixel_data7[115]), // Write Data
3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27		.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]), .DA42(ipixel_data7[42]), .DA46(ipixel_data7[46]), .DA50(ipixel_data7[50]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 6 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[83]), // Write Data 10 DA88(ipixel_data7[87]), // Write Data 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[85]), 12 DA91(ipixel_data7[88]), DA93(ipixel_data7[89]), 13 DA92(ipixel_data7[98]), // Write Data 14 DA95(ipixel_data7[91]), // Write Data 15 DA96(ipixel_data7[92]), DA97(ipixel_data7[93]), 16 DA96(ipixel_data7[96]), // Write Data 17 DA96(ipixel_data7[96]), DA97(ipixel_data7[97]), 18 DA106(ipixel_data7[100]), DA101(ipixel_data7[101]), 18 DA103(ipixel_data7[103]), // Write Data 19 DA104(ipixel_data7[104]), DA105(ipixel_data7[105]), 20 DA107(ipixel_data7[107]), // Write Data 21 DA108(ipixel_data7[108]), DA108(ipixel_data7[109]), 22 DA116(ipixel_data7[111]), // Write Data 23 DA112(ipixel_data7[112]), DA113(ipixel_data7[113]), DA114(ipixel_data7[114]), 24 DA115(ipixel_data7[115]), // Write Data 25 DA116(ipixel_data7[115]), // Write Data 26 DA119(ipixel_data7[116]), DA117(ipixel_data7[117]), DA118(ipixel_data7[118]), 27 DA120(ipixel_data7[119]), // Write Data 28 DA116(ipixel_data7[112]), // Write Data 29 DA116(ipixel_data7[112]), // Write Data 20 DA116(ipixel_data7[112]), // Write Data 21 DA116(ipixel_data7[112]), // Write Data 22 DA116(ipixel_data7[112]), // Write Data 23 DA116(ipixel_data7[112]), // Write Data
3 4 4 5 6 6 7 7 8 9 10 11 12 13 3 14 4 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28 29 29		.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[34]), .DA42(ipixel_data7[42]), .DA46(ipixel_data7[42]), .DA50(ipixel_data7[50]), .DA50(ipixel_data7[50]), .DA54(ipixel_data7[50]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 6 DA79(ipixel_data7[76]), DA77(ipixel_data7[77]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), DA82(ipixel_data7[82]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), 10 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), 11 DA88(ipixel_data7[91]), // Write Data 12 DA92(ipixel_data7[91]), // Write Data 13 DA92(ipixel_data7[91]), DA93(ipixel_data7[93]), 14 DA95(ipixel_data7[95]), // Write Data 15 DA96(ipixel_data7[96]), DA97(ipixel_data7[97]), 16 DA99(ipixel_data7[99]), // Write Data 17 DA100(ipixel_data7[100]), DA101(ipixel_data7[101]), 18 DA103(ipixel_data7[104]), DA105(ipixel_data7[105]), DA106(ipixel_data7[102]), 20 DA104(ipixel_data7[104]), DA105(ipixel_data7[105]), DA106(ipixel_data7[106]), 21 DA104(ipixel_data7[118]), DA109(ipixel_data7[107]), DA110(ipixel_data7[107]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA111(ipixel_data7[117]), DA112(ipixel_data7[117]), DA112(ipixel_data7[117]), DA112(ipixel_data7[117]), DA112(ipixel_data7[117]), DA112(ipixel_data7[118]), DA112(ipixel_data7[118]), DA112(ipixel_data7[118]), DA112(ipixel_data7[118]), DA112(ipixel_data7[118]), DA112(ipixel_data7[118]), DA112(ipixel_data7[124]), DA112(ipixel_data7[125]), DA112(ipixel_data7[126]), DA112(ipixel_data7[125]), DA1126(ipixel_data7[126]), DA1126(ipixel_data7[
3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 13 14 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 31		.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]), .DA42(ipixel_data7[42]), .DA50(ipixel_data7[46]), .DA50(ipixel_data7[50]), .DA54(ipixel_data7[54]), .DA58(ipixel_data7[54]), .DA58(ipixel_data7[58]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 6 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[83]), // Write Data 10 DA84(ipixel_data7[87]), // Write Data 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), 12 DA91(ipixel_data7[88]), DA93(ipixel_data7[89]), 13 DA92(ipixel_data7[98]), // Write Data 14 DA95(ipixel_data7[92]), DA93(ipixel_data7[93]), 15 DA96(ipixel_data7[92]), DA97(ipixel_data7[93]), 16 DA96(ipixel_data7[99]), // Write Data 17 DA96(ipixel_data7[99]), // Write Data 18 DA104(ipixel_data7[100]), DA101(ipixel_data7[101]), 19 DA103(ipixel_data7[101]), // Write Data 19 DA103(ipixel_data7[104]), DA105(ipixel_data7[105]), 20 DA107(ipixel_data7[107]), // Write Data 21 DA108(ipixel_data7[108]), DA109(ipixel_data7[109]), 22 DA111(ipixel_data7[111]), // Write Data 23 DA112(ipixel_data7[112]), DA113(ipixel_data7[113]), DA110(ipixel_data7[114]), 24 DA115(ipixel_data7[115]), // Write Data 25 DA116(ipixel_data7[115]), // Write Data 26 DA119(ipixel_data7[115]), // Write Data 27 DA119(ipixel_data7[119]), // Write Data 28 DA119(ipixel_data7[119]), // Write Data 29 DA119(ipixel_data7[119]), // Write Data 20 DA119(ipixel_data7[115]), // Write Data 21 DA119(ipixel_data7[115]), // Write Data 22 DA119(ipixel_data7[115]), // Write Data 23 DA119(ipixel_data7[115]), // Write Data 24 DA119(ipixel_data7[115]), // Write Data 25 DA119(ipixel_data7[115]), // Write Data 26 DA119(ipixel_data7[115]), // Write Data
3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	.DA7(ipixel_data7[7]), // Write Data .DA8(ipixel_data7[1]), // Write Data .DA8(ipixel_data7[1]), // Write Data .DA12(ipixel_data7[1]), // Write Data .DA12(ipixel_data7[12]), .DA13(ipixel_data7[13]), .DA15(ipixel_data7[13]), .DA15(ipixel_data7[13]), // Write Data .DA16(ipixel_data7[16]), // Write Data .DA20(ipixel_data7[20]), .DA17(ipixel_data7[21]), .DA19(ipixel_data7[21]), // Write Data .DA24(ipixel_data7[24]), .DA25(ipixel_data7[25]), .DA27(ipixel_data7[27]), // Write Data .DA28(ipixel_data7[28]), .DA25(ipixel_data7[25]), .DA27(ipixel_data7[27]), // Write Data .DA36(ipixel_data7[32]), .DA33(ipixel_data7[32]), .DA35(ipixel_data7[35]), // Write Data .DA36(ipixel_data7[36]), .DA37(ipixel_data7[37]), .DA39(ipixel_data7[40]), .DA31(ipixel_data7[41]), .DA41(ipixel_data7[41]), .DA44(ipixel_data7[44]), .DA45(ipixel_data7[44]), .DA45(ipixel_data7[48]), // Write Data .DA44(ipixel_data7[48]), // Write Data .DA48(ipixel_data7[48]), .DA49(ipixel_data7[49]), .DA51(ipixel_data7[45]), .DA57(ipixel_data7[55]), // Write Data .DA56(ipixel_data7[55]), // Write Data .DA56(ipixel_data7[56]), .DA57(ipixel_data7[57]), .DA55(ipixel_data7[56]), .DA57(ipixel_data7[57]), .DA56(ipixel_data7[56]), // Write Data .DA60(ipixel_data7[60]), .DA51(ipixel_data7[61]), .DA63(ipixel_data7[60]), // Write Data	.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]), .DA42(ipixel_data7[42]), .DA46(ipixel_data7[42]), .DA50(ipixel_data7[50]), .DA50(ipixel_data7[50]), .DA58(ipixel_data7[58]), .DA58(ipixel_data7[58]), .DA62(ipixel_data7[62]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), DA73(ipixel_data7[73]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), DA77(ipixel_data7[77]), 6 DA79(ipixel_data7[76]), DA77(ipixel_data7[77]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), DA82(ipixel_data7[82]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), 10 DA87(ipixel_data7[88]), DA89(ipixel_data7[89]), 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), 12 DA91(ipixel_data7[91]), // Write Data 13 DA92(ipixel_data7[91]), // Write Data 14 DA92(ipixel_data7[91]), // Write Data 15 DA96(ipixel_data7[96]), DA97(ipixel_data7[97]), 16 DA99(ipixel_data7[99]), // Write Data 17 DA106(ipixel_data7[100]), DA101(ipixel_data7[101]), 18 DA103(ipixel_data7[100]), DA101(ipixel_data7[101]), 19 DA104(ipixel_data7[104]), DA105(ipixel_data7[105]), 20 DA104(ipixel_data7[107]), // Write Data 21 DA108(ipixel_data7[108]), DA109(ipixel_data7[109]), DA106(ipixel_data7[101]), 22 DA111(ipixel_data7[111]), DA113(ipixel_data7[113]), DA114(ipixel_data7[114]), 23 DA112(ipixel_data7[115]), // Write Data 25 DA116(ipixel_data7[116]), DA117(ipixel_data7[117]), DA118(ipixel_data7[118]), 26 DA116(ipixel_data7[116]), DA117(ipixel_data7[117]), DA118(ipixel_data7[118]), 27 DA106(ipixel_data7[116]), // Write Data 28 DA124(ipixel_data7[121]), // Write Data 29 DA124(ipixel_data7[124]), // Write Data 29 DA124(ipixel_data7[124]), // Write Data 20 DA125(ipixel_data7[125]), // Write Data 21 DA126(ipixel_data7[124]), // DA125(ipixel_data7[125]), // DA126(ipixel_data7[126]), 28 DA126(ipixel_data7[127]), // Write Data
3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 13 14 14 15 16 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 31		.DA10(ipixel_data7[10]), .DA14(ipixel_data7[14]), .DA18(ipixel_data7[18]), .DA22(ipixel_data7[22]), .DA26(ipixel_data7[26]), .DA30(ipixel_data7[30]), .DA34(ipixel_data7[34]), .DA38(ipixel_data7[38]), .DA42(ipixel_data7[42]), .DA46(ipixel_data7[42]), .DA50(ipixel_data7[50]), .DA50(ipixel_data7[50]), .DA58(ipixel_data7[58]), .DA58(ipixel_data7[58]), .DA62(ipixel_data7[62]),	2 DA71(ipixel_data7[71]), // Write Data 3 DA72(ipixel_data7[72]), // DA73(ipixel_data7[73]), 4 DA75(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[75]), // Write Data 5 DA76(ipixel_data7[76]), // DA77(ipixel_data7[77]), 6 DA79(ipixel_data7[76]), // DA77(ipixel_data7[77]), 7 DA80(ipixel_data7[80]), DA81(ipixel_data7[81]), // DA82(ipixel_data7[82]), 8 DA83(ipixel_data7[83]), // Write Data 9 DA84(ipixel_data7[84]), DA85(ipixel_data7[85]), // DA87(ipixel_data7[87]), // Write Data 11 DA88(ipixel_data7[88]), DA89(ipixel_data7[89]), DA90(ipixel_data7[90]), 12 DA91(ipixel_data7[91]), // Write Data 13 DA92(ipixel_data7[91]), // Write Data 14 DA95(ipixel_data7[91]), // Write Data 15 DA96(ipixel_data7[96]), DA97(ipixel_data7[97]), DA98(ipixel_data7[98]), 16 DA96(ipixel_data7[100]), DA101(ipixel_data7[101]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA103(ipixel_data7[100]), DA104(ipixel_data7[100]), DA104(ipixel_data7[100]), DA105(ipixel_data7[100]), DA105(ipixel_data7[100]), DA106(ipixel_data7[100]), DA106(ipixel_data7[100]), DA109(ipixel_data7[100]), DA106(ipixel_data7[100]), DA106(ipix

```
.TDA52(ipixel_data7[52]), .TDA53(i
.TDA55(ipixel_data7[55]), // Write Test Data
            .TWEA(vss), // Test write enable
                                                                                                                                                                                                                 .TDA53(ipixel_data7[53]),
                                                                                                                                                                                                                                                        .TDA54(ipixel_data7[54]),
2
            .TMEA(vss), // Test memory enable
                                                                                                                                                                     .TDA56(ipixel_data7[56]), .TDA57(ipixel_data7[57]), .TDA59(ipixel_data7[59]), // Write Test Data
                                                                                                                                                                                                                                                         .TDA58(ipixel_data7[58]),
        .TADRA0(iwrite_addr[0]), .TADRA1(iw.TADRA3(iwrite_addr[3]), // Write Test Address
                                                 .TADRA1(iwrite_addr[1]),
                                                                                         .TADRA2(iwrite addr[2])
                                                                                                                                                                      .TDA60(ipixel_data7[60]), .TDA61(ipixel_data7[61]), .TDA63(ipixel_data7[63]), // Write Test Data
                                                                                                                                                                                                                                                         .TDA62(ipixel_data7[62]),
            .TADRA4(iwrite_addr[4]), .TADRA5(iwrite_addr[5]), .TADRA6(iwrite_addr[6]), // Write
                                                                                                                                                                     .TDA64(ipixel_data7[64]), .TDA65(ipixel_data7[65]), .TDA67(ipixel_data7[67]), // Write Test Data
                                                                                                                                                                                                                                                         .TDA66(ipixel data7[66]),
        .TDA1(ipixel_data7[0]), .TDA1(ipixel_data7[1]), .TDA3(ipixel_data7[3]), // Write Test Data
                                                                                             .TDA2(ipixel_data7[2]),
                                                                                                                                                                                                              .TDA69(ipixel_data7[69]),
                                                                                                                                                                          .TDA68(ipixel_data7[68]).
                                                                                                                                                                                                                                                        .TDA70(ipixel_data7[70]).
        .TDA4(ipixel_data7[4]), .TDA5
.TDA7(ipixel_data7[7]), // Write Test Data
                                                   .TDA5(ipixel_data7[5]),
                                                                                             .TDA6(ipixel_data7[6]).
                                                                                                                                                                     .TDA71(ipixel_data7[71]), // Write Test Data
10
                                                                                                                                                                     .TDA72(ipixel_data7[72]), .TDA73(i
.TDA75(ipixel_data7[75]), // Write Test Data
                                                                                                                                                                                                               .TDA73(ipixel_data7[73]),
                                                                                                                                                                                                                                                         .TDA74(ipixel data7[74]),
                                                   .TDA9(ipixel_data7[9]),
        .TDA8(ipixel_data7[8]), .TDA9(i
.TDA11(ipixel_data7[11]), // Write Test Dat
                                                                                           .TDA10(ipixel data7[10]),
                                                                                                                                                                     .TDA76(ipixel_data7[76]), .TDA77(ipixel_data7[77]), .TDA79(ipixel_data7[79]), // Write Test Data
                                                                                                                                                                                                                                                         .TDA78(ipixel data7[78]),
        .TDA12(ipixel_data7[12]), .TDA13(ip
.TDA15(ipixel_data7[15]), // Write Test Data
                                                 .TDA13(ipixel_data7[13]),
                                                                                          .TDA14(ipixel_data7[14]),
                                                                                                                                                                     .TDA80(ipixel_data7[80]), .TDA81(ipixel_data7[83]), // Write Test Data
                                                                                                                                                                                                               .TDA81(ipixel_data7[81]),
                                                                                                                                                                                                                                                         .TDA82(ipixel_data7[82]),
        .TDA16(ipixel_data7[16]), .TDA17(ipixel_data7[17]), .TDA19(ipixel_data7[19]), // Write Test Data
                                                                                           .TDA18(ipixel data7[18]),
                                                                                                                                                                                                              .TDA85(ipixel_data7[85]),
                                                                                                                                                                     .TDA84(ipixel_data7[84]), .TDA85(i
.TDA87(ipixel_data7[87]), // Write Test Data
                                                                                                                                                                                                                                                        .TDA86(ipixel_data7[86]).
17
18
        .TDA20(ipixel_data7[20]), .TDA21(ipixel_data7[21]), .TDA23(ipixel_data7[23]), // Write Test Data
                                                                                          .TDA22(ipixel data7[22]),
                                                                                                                                                                     .TDA88(ipixel_data7[88]), .TDA89(i
.TDA91(ipixel_data7[91]), // Write Test Data
                                                                                                                                                                                                                .TDA89(ipixel_data7[89]),
                                                                                                                                                                                                                                                        .TDA90(ipixel data7[90]),
       .TDA24(ipixel_data7[24]), .TDA25(ipixel_data7[25]), .TDA27(ipixel_data7[27]), // Write Test Data
19
20
                                                                                          .TDA26(ipixel data7[26]),
                                                                                                                                                                     .TDA92(ipixel_data7[92]), .TDA93(ipixel_data7[93]), .TDA95(ipixel_data7[95]), // Write Test Data
                                                                                                                                                                                                                                                         .TDA94(ipixel_data7[94]),
        .TDA28(ipixel_data7[28]), .TDA29(ipixel_data7[29]), .TDA31(ipixel_data7[31]), // Write Test Data
                                                                                          .TDA30(ipixel data7[30]),
22
                                                                                                                                                                     .TDA96(ipixel_data7[96]), .TDA97(ipixel_data7[97]), .TDA99(ipixel_data7[99]), // Write Test Data
                                                                                                                                                                                                                                                        .TDA98(ipixel_data7[98]),
        .TDA32(ipixel_data7[32]), .TDA33(ipixel_data7[33]), .TDA35(ipixel_data7[35]), // Write Test Data
                                                                                          .TDA34(ipixel_data7[34]),
23
24
                                                                                                                                                                     .
TDA100(ipixel_data7[100]), .
TDA101(ipixel_data7[101]), .
TDA103(ipixel_data7[103]), // Write Test Data
        .TDA36(ipixel_data7[36]), .TDA37(ipixel_data7[37]), .TDA39(ipixel_data7[39]), // Write Test Data
25
26
                                                                                          .TDA38(ipixel_data7[38]),
                                                                                                                                                                     .
TDA104(ipixel_data7[104]), .
TDA105(ipixel_data7[105]), .
TDA107(ipixel_data7[107]), // Write Test Data
        .TDA40(ipixel_data7[40]), .TDA41(i
.TDA43(ipixel_data7[43]), // Write Test Data
                                                .TDA41(ipixel_data7[41]),
27
28
                                                                                          .TDA42(ipixel data7[42]),
                                                                                                                                                                     . TDA108(ipixel_data7[108]), . TDA109(ipixel_data7[109]), . TDA110(ipixel_data7[110]), . TDA111(ipixel_data7[111]), . // Write Test Data
                                                 .TDA45(ipixel_data7[45]),
        .TDA44(ipixel_data7[44]), .TDA45(i
.TDA47(ipixel_data7[47]), // Write Test Data
                                                                                          .TDA46(ipixel data7[46]),
                                                                                                                                                                     .TDA112(ipixel_data7[112]), .TDA113(ipixel_data7[113]), .TDA114(ipixel_data7[114]), .TDA115(ipixel_data7[115]), // Write Test Data
        .TDA48(ipixel_data7[48]), .TDA49(ipixel_data7[49]), .TDA51(ipixel_data7[51]), // Write Test Data
                                                                                          .TDA50(ipixel_data7[50]),
                                                         Page 69 of 84
                                                                                                                                                                                                                       Page 70 of 84
                                                                                         Ex. 2113 - export_buffers.v
                                                                                                                                                                                                                                                        Ex. 2113 - export_buffers.v
       .
TDA116(ipixel_data7[116]), .
TDA117(ipixel_data7[117]), .
TDA118(ipixel_data7[118]), .
TDA119(ipixel_data7[119]), // Write Test Data
                                                                                                                                                                              else
                                                                                                                                                                               begin
        . TDA120(ipixel\_data7[120]), \quad . TDA121(ipixel\_data7[121]), \quad . TDA122(ipixel\_data7[122]), \quad . TDA123(ipixel\_data7[123]), \quad // Write Test Data
                                                                                                                                                                                 q\_sp\_bank\_sel <= sp\_bank\_sel;
        .
TDA124(ipixel_data7[124]), .
TDA125(ipixel_data7[125]), .
TDA126(ipixel_data7[126]), .
TDA127(ipixel_data7[127]), .
Write Test Data
                                                                                                                                                                                 q_sp_bank_sel0 <= sp_bank_sel0;
                                                                                                                                                                                  q_sp_bank_sel1 <= sp_bank_sel1;
           //READ TEST SIGNALS
                                                                                                                                                                                  q_sp_bank_sel2 <= sp_bank_sel2;
            .BISTEB(vss),
                                                                                                                                                                                  q\_sp\_bank\_sel3 \mathrel{<=} sp\_bank\_sel3;
            .TOEB(vss),
                                                                                                                                                                                end // else: !if()
                                                                                                                                                                         end // always @ (posedge sclk)
        . TADRB0(q2\_read\_addr[0]), \quad . TADRB1(q2\_read\_addr[1]), \quad . TADRB2(q2\_read\_addr[2]), \\ . TADRB3(q2\_read\_addr[3]), \quad // Read Test Address
11
12
                                                                                                                                                              10
                                                                                                                                                             11
            .TADRB4(q2 read addr[4]), .TADRB5(q2 read addr[5]), .TADRB6(q2 read addr[6]), //
14
                                                                                                                                                              12
15
            .AWTB(vss)
                                                                                                                                                              13
16
                                                                                                                                                              14
17
18
        'endif // !'ifdef USE BEHAVE MEM
                                                                                                                                                                        always @(/*AUTOSENSE*/buff0_out or buff1_out or buff2_out
19
                                                                                                                                                              17
                                                                                                                                                                                 or buff3_out or q_phase_rb0)
20
                                                                                                                                                              18
                                                                                                                                                                         begin
21
                                                                                                                                                              19
                                                                                                                                                                              case(q phase rb0)
22
         always @(posedge sclk)
                                                                                                                                                              20
                                                                                                                                                                               2'b00:bank0 data0 = buff0 out;
23
          begin
                                                                                                                                                             21
                                                                                                                                                                               2'b01:bank0 data0 = buff1 out;
24
                                                                                                                                                             22
                                                                                                                                                                               2'b10:bank0 data0 = buff2 out:
25
                 begin
                                                                                                                                                             23
                                                                                                                                                                               2'b11:bank0 data0 = buff3 out:
26
                   q\_sp\_bank\_sel \mathrel{<=} 1"b0;
                                                                                                                                                             24
                                                                                                                                                                                default:bank0 data0 = buff0 out;
27
                 end
                                                                                                                                                                              endcase // case(q phase rb0)
                                                          Page 71 of 84
                                                                                                                                                                                                                       Page 72 of 84
                                                                                         Ex. 2113 - export buffers.v
                                                                                                                                                                                                                                                       Ex. 2113 - export buffers.v
```

```
end // always @ (...
                                                                                                                                    default:bank0 data2 = buff2 out;
 1
 2
                                                                                                                                   endcase // case(q_phase_rb2)
 3
       //buffer 1 4x1 mux
                                                                                                                               end // always @ (...
       always @(/*AUTOSENSE*/buff0_out or buff1_out or buff2_out
 5
              or buff3_out or q_phase_rb1)
        begin
                                                                                                                              //buffer 3 4x1 mux
            case(q_phase_rb1)
                                                                                                                              always @(/*AUTOSENSE*/buff0_out or buff1_out or buff2_out
            2'b00:bank0_data1 = buff0_out;
                                                                                                                                     or buff3_out or q_phase_rb3)
             2'b01:bank0_data1 = buff1_out;
10
             2'b10:bank0_data1 = buff2_out;
                                                                                                                      10
                                                                                                                                   case(q_phase_rb3)
11
             2'b11:bank0_data1 = buff3_out;
                                                                                                                                   2'b00:bank0_data3 = buff0_out;
12
             default:bank0_data1 = buff0_out;
                                                                                                                                   2'b01:bank0_data3 = buff1_out;
13
            endcase // case(q_phase_rb1)
                                                                                                                                  2'b10:bank0_data3 = buff2_out;
14
        end // always @ (...
                                                                                                                                   2'b11:bank0_data3 = buff3_out;
15
                                                                                                                                   default:bank0_data3 = buff0_out;
                                                                                                                      15
16
                                                                                                                      16
                                                                                                                                   endcase // case(q phase rb3)
17
        //buffer 2 4x1 mux
                                                                                                                      17
                                                                                                                              end // always @ (...
       always @(/*AUTOSENSE*/buff0 out or buff1 out or buff2 out
                                                                                                                      18
18
19
              or buff3_out or q_phase_rb2)
                                                                                                                      19
                                                                                                                      20
                                                                                                                              //buffer 4 4x1 mux
20
        begin
21
                                                                                                                              always @(/*AUTOSENSE*/buff4 out or buff5 out or buff6 out
            case(q_phase_rb2)
                                                                                                                      21
             2'b00:bank0 data2 = buff0 out;
22
                                                                                                                      22
                                                                                                                                     or buff7_out or q_phase_rb0)
23
             2'b01:bank0 data2 = buff1 out;
                                                                                                                      23
24
             2'b10:bank0_data2 = buff2_out;
                                                                                                                      24
                                                                                                                                   case(q_phase_rb0)
             2'b11:bank0_data2 = buff3_out;
                                                                                                                      25
                                                                                                                                    2'b00:bank1_data0 = buff4_out;
                                           Page 73 of 84
                                                                                                                                                                 Page 74 of 84
                                                                   Ex. 2113 - export_buffers.v
                                                                                                                                                                                          Ex. 2113 - export_buffers.v
             2'b01:bank1_data0 = buff5_out;
                                                                                                                                  case(q phase rb2)
             2'b10:bank1_data0 = buff6_out;
 2
                                                                                                                                   2'b00:bank1 data2 = buff4 out;
             2'b11:bank1 data0 = buff7 out:
                                                                                                                                   2'b01:bank1 data2 = buff5 out:
             default:bank1 data0 = buff4 out:
                                                                                                                                   2'b10:bank1 data2 = buff6 out:
            endcase // case(q_phase_rb0)
                                                                                                                                    2'b11:bank1 data2 = buff7 out;
        end // always @ (...
                                                                                                                                    default:bank1 data2 = buff4 out;
                                                                                                                                   endcase // case(q_phase_rb2)
                                                                                                                              end // always @ (...
        //buffer 5 4x1 mux
       always @(/*AUTOSENSE*/buff4_out or buff5_out or buff6_out
10
                                                                                                                      10
                                                                                                                              //buffer 7 4x1 mux
11
              or buff7_out or q_phase_rb1)
                                                                                                                      11
                                                                                                                              always @(/*AUTOSENSE*/buff4_out or buff5_out or buff6_out
12
                                                                                                                      12
                                                                                                                                     or buff7_out or q_phase_rb3)
13
            case(q_phase_rb1)
                                                                                                                      13
14
            2'b00:bank1_data1 = buff4_out;
                                                                                                                      14
                                                                                                                                  case(q_phase_rb3)
15
            2'b01:bank1_data1 = buff5_out;
                                                                                                                                 2'b00:bank1_data3 = buff4_out;
16
             2'b10:bank1_data1 = buff6_out;
                                                                                                                                  2'b01:bank1_data3 = buff5_out;
17
             2'b11:bank1_data1 = buff7_out;
                                                                                                                                  2'b10:bank1_data3 = buff6_out;
18
             default:bank1_data1 = buff4_out;
                                                                                                                                   2'b11:bank1_data3 = buff7_out;
19
            endcase // case(q_phase_rb1)
                                                                                                                                    default:bank1_data3 = buff4_out;
        end // always @ (...
                                                                                                                                  endcase // case(q_phase_rb3)
20
                                                                                                                      20
21
                                                                                                                              end // always @ (...
                                                                                                                      21
22
        //buffer 6 4x1 mux
                                                                                                                      22
23
       always @(/*AUTOSENSE*/buff4 out or buff5 out or buff6 out
                                                                                                                      23
                                                                                                                             //clipper data from bank0 (sp0)
24
                                                                                                                      24
                                                                                                                              always @(/*AUTOSENSE*/buff0 out or buff1_out or buff2_out
              or buff7_out or q_phase_rb2)
25
        begin
                                                                                                                      25
                                                                                                                                     or buff3 out or q phase clipp)
                                           Page 75 of 84
                                                                                                                                                                 Page 76 of 84
                                                                   Ex. 2113 - export_buffers.v
                                                                                                                                                                                         Ex. 2113 - export buffers.v
```

```
begin
 2
            case(q\_phase\_clipp)
             2'b00:bank0_clipp_data = buff0_out;
                                                                                                                                  reg [127:0] clipp_data;
              2'b01:bank0_clipp_data = buff1_out;
              2'b10:bank0\_clipp\_data = buff2\_out;
             2'b11:bank0_clipp_data = buff3_out;
                                                                                                                                  //final mux selecting between left and right banks of the export buffers (sp0 vs. sp1)
              default:bank0_clipp_data = buff0_out;
                                                                                                                                  always @(/*AUTOSENSE*/bank0_clipp_data or bank1_clipp_data
            endcase // case(q_phase_clipp)
                                                                                                                                         or q_sp_bank_sel)
        end // always @ (...
                                                                                                                                   begin
10
                                                                                                                           10
                                                                                                                                       case(q\_sp\_bank\_sel)
11
       //clipper data from bank1 (sp1)
                                                                                                                                        1'b0:clipp_data=bank0_clipp_data;
12
       always @(/*AUTOSENSE*/buff4_out or buff5_out or buff6_out
                                                                                                                           12
                                                                                                                                         1'b1:clipp_data=bank1_clipp_data;
13
                                                                                                                           13
               or buff7 out or q phase clipp)
                                                                                                                                       endcase // case(q sp bank sel)
14
                                                                                                                           14
        begin
15
                                                                                                                          15
            case(q phase clipp)
              2'b00:bank1_clipp_data = buff4_out;
16
                                                                                                                           16
                                                                                                                                   //four outputs ...one for each RB
17
              2'b01:bank1_clipp_data = buff5_out;
                                                                                                                          17
                                                                                                                                  always @(/*AUTOSENSE*/bank0_data0 or bank1_data0 or q_sp_bank_sel0)
18
              2'b10:bank1_clipp_data = buff6_out;
                                                                                                                          18
19
              2'b11:bank1_clipp_data = buff7_out;
                                                                                                                          19
                                                                                                                                   begin
              default:bank0 clipp data = buff4 out;
20
                                                                                                                          20
                                                                                                                                       case(q sp bank sel0)
21
            endcase // case(q_phase_clipp)
                                                                                                                          21
                                                                                                                                        1'b0:rb0 data=bank0 data0
22
         end // always @ (...
                                                                                                                          22
                                                                                                                                        1'b1:rb0 data=bank1 data0:
23
                                                                                                                          23
                                                                                                                                       endcase // case(q_sp_bank_sel0)
24
                                                                                                                          24
25
                                                                                                                                  always @ (/*AUTOSENSE*/bank0\_data1 \ or \ bank1\_data1 \ or \ q\_sp\_bank\_sel1)
                                             Page 77 of 84
                                                                                                                                                                       Page 78 of 84
                                                                      Ex. 2113 - export_buffers.v
                                                                                                                                                                                                Ex. 2113 - export_buffers.v
        begin
                                                                                                                                         or q phase clipp)
 2
                                                                                                                           2
            case(q sp bank sell)
                                                                                                                                   begin
              1'b0:rb1 data=bank0 data1;
                                                                                                                                       case(q_phase_clipp)
              l'b1:rb1 data=bank1 data1:
                                                                                                                                        2'b00:clipp data valid = q0 read valid clipp;
            endcase // case(q_sp_bank_sel1)
                                                                                                                                        2'b01:clipp_data_valid = q1_read_valid_clipp;
                                                                                                                                        2'b10:clipp_data_valid = q2_read_valid_clipp;
 7
       always @ (/*AUTOSENSE*/bank0\_data2 \ or \ bank1\_data2 \ or \ q\_sp\_bank\_sel2)
                                                                                                                                        2"b11:clipp\_data\_valid = q3\_read\_valid\_clipp;
 8
                                                                                                                                        default:clipp\_data\_valid = q0\_read\_valid\_clipp;
            case(q_sp_bank_sel2)
                                                                                                                                       endcase // case(q_phase_clipp)
10
             1'b0:rb2_data=bank0_data2;
                                                                                                                           10
                                                                                                                                   end // always @ (...
11
              1'b1:rb2_data=bank1_data2;
                                                                                                                          11
12
            endcase // case(q_sp_bank_sel2)
                                                                                                                          12
13
                                                                                                                                  //generating the data valid for the export buffer read over four cycles
14
        always @(/*AUTOSENSE*/bank0_data3 or bank1_data3 or q_sp_bank_sel3)
                                                                                                                          14
                                                                                                                                  always @(/*AUTOSENSE*/q0_read_valid_rb0 or q1_read_valid_rb0
15
                                                                                                                          15
                                                                                                                                         or q2_read_valid_rb0 or q3_read_valid_rb0 or q_phase_rb0)
16
                                                                                                                           16
            case(q\_sp\_bank\_sel3)
17
              1'b0:rb3_data=bank0_data3;
                                                                                                                           17
                                                                                                                                       case(q_phase_rb0)
18
              1'b1:rb3_data=bank1_data3;
                                                                                                                           18
                                                                                                                                        2'b00:rb0_data_valid = q0_read_valid_rb0;
19
            endcase // case(q_sp_bank_sel3)
                                                                                                                                        2'b01:rb0_data_valid = q1_read_valid_rb0;
20
                                                                                                                                        2'b10:rb0 data valid = q2 read valid rb0;
        end
                                                                                                                          20
21
                                                                                                                          21
                                                                                                                                        2'b11:rb0 data valid = q3 read valid rb0;
22
                                                                                                                          22
                                                                                                                                        default:rb0 data valid = a0 read valid rb0:
23
                                                                                                                          23
       //generating the data valid for the export buffer read over four cycles
                                                                                                                                       endcase // case(q phase rb0)
24
       always @(/*AUTOSENSE*/q0_read_valid_clipp or q1_read_valid_clipp
                                                                                                                          24
                                                                                                                                   end // always @ (...
2.5
               or q2_read_valid_clipp or q3_read_valid_clipp
                                             Page 79 of 84
                                                                                                                                                                       Page 80 of 84
                                                                      Ex. 2113 - export buffers.v
                                                                                                                                                                                                Ex. 2113 - export buffers.v
```

```
end // always @ (...
 2
             //generating the data valid for the export buffer read over four cycles
                                                                                                                                                                                                                              2
 3
             always @(/*AUTOSENSE*/q0_read_valid_rb1 or q1_read_valid_rb1
 4
                          or q2_read_valid_rb1 or q3_read_valid_rb1 or q_phase_rb1)
                                                                                                                                                                                                                                         //generating the data valid for the export buffer read over four cycles
 5
               begin
                                                                                                                                                                                                                                          always @(/*AUTOSENSE*/q0_read_valid_rb3 or q1_read_valid_rb3
                      case(q\_phase\_rb1)
                                                                                                                                                                                                                                                       or q2_read_valid_rb3 or q3_read_valid_rb3 or q_phase_rb3)
                       2'b00:rb1_data_valid = q0_read_valid_rb1;
                        2'b01:rb1_data_valid = q1_read_valid_rb1;
                                                                                                                                                                                                                                                   case(q_phase_rb3)
                        2'b10:rb1_data_valid = q2_read_valid_rb1;
                                                                                                                                                                                                                                                     2'b00:rb3_data_valid = q0_read_valid_rb3;
10
                       2'b11:rb1_data_valid = q3_read_valid_rb1;
                                                                                                                                                                                                                                                    2'b01:rb3_data_valid = q1_read_valid_rb3;
11
                        default:rb1_data_valid = q0_read_valid_rb1;
                                                                                                                                                                                                                                                   2'b10:rb3_data_valid = q2_read_valid_rb3;
12
                      endcase // case(q_phase_rb1)
                                                                                                                                                                                                                                                     2'b11:rb3_data_valid = q3_read_valid_rb3;
13
               end // always @ (...
                                                                                                                                                                                                                                                     default:rb3_data_valid = q0_read_valid_rb3;
14
                                                                                                                                                                                                                                                   endcase // case(q_phase_rb3)
15
                                                                                                                                                                                                                                           end // always @ (...
                                                                                                                                                                                                                            15
              //generating the data valid for the export buffer read over four cycles
             always @(/*AUTOSENSE*/q0_read_valid_rb2 or q1_read_valid_rb2
16
17
                                                                                                                                                                                                                            17
                          or q2 read valid rb2 or q3 read valid rb2 or q phase rb2)
18
                                                                                                                                                                                                                            18
               begin
19
                      case(q_phase_rb2)
                                                                                                                                                                                                                            19
                                                                                                                                                                                                                                         reg [127:0] q_clipp_data;
                        2'b00:rb2_data_valid = q0_read_valid_rb2;
20
                                                                                                                                                                                                                            20
                                                                                                                                                                                                                                                               q_clipp_data_valid;
21
                        2'b01:rb2 data valid = q1 read valid rb2;
                                                                                                                                                                                                                           21
                                                                                                                                                                                                                                                         \label{eq:continuous} \verb|q_rb0_data_valid|, \verb|q_rb1_data_valid|, \verb|q_rb2_data_valid|, \verb|q_rb3_data_valid|, \|q_rb3_data_valid|, \|q_rb
                        2'b10:rb2_data_valid = q2_read_valid_rb2;
22
                                                                                                                                                                                                                            22
23
                        2'b11:rb2_data_valid = q3_read_valid_rb2;
                                                                                                                                                                                                                           23
                                                                                                                                                                                                                                          always @(posedge sclk)
24
                        default:rb2_data_valid = q0_read_valid_rb2;
                                                                                                                                                                                                                            24
                       endcase // case(q_phase_rb2)
                                                                                                                                                                                                                            25
                                                                                                                                                                                                                                                   q_rb0_data <= rb0_data;
                                                                                Page 81 of 84
                                                                                                                                                                                                                                                                                                             Page 82 of 84
                                                                                                                              Ex. 2113 - export_buffers.v
                                                                                                                                                                                                                                                                                                                                                          Ex. 2113 - export_buffers.v
                      q_rb0_data_valid <= rb0_data_valid;
  2
                      q rb1 data <= rb1 data;
                      q rb1 data valid <= rb1 data valid;
                      q rb2 data <= rb2 data;
                      q_rb2_data_valid <= rb2_data_valid;
                      q rb3 data <= rb3 data;
                       q\_rb3\_data\_valid <= rb3\_data\_valid;
                       q\_clipp\_data \mathrel{<=} clipp\_data;
                       q\_clipp\_data\_valid <= clipp\_data\_valid;
10
                                                                                                                                                                                                                             10
11
                                                                                                                                                                                                                             11
12
             assign\ orb0\_data = q\_rb0\_data;
                                                                                                                                                                                                                             12
13
             assign orb1_data = q_rb1_data;
                                                                                                                                                                                                                             13
14
             assign orb2_data = q_rb2_data;
                                                                                                                                                                                                                             14
             assign orb3_data = q_rb3_data;
              assign oclipp_data = q_clipp_data;
17
18
             assign orb0_data_valid = q_rb0_data_valid;
19
             assign orbl_data_valid = q_rbl_data_valid;
20
             assign orb2 data valid = q rb2 data valid;
21
             assign orb3 data valid = q rb3 data valid;
22
             assign oclipp_data_valid = q_clipp_data_valid;
23
24
          endmodule // export buffers
25
                                                                                 Page 83 of 84
                                                                                                                                                                                                                                                                                                             Page 84 of 84
                                                                                                                                                                                                                                                                                                                                                         Ex. 2113 - export_buffers.v
                                                                                                                              Ex. 2113 - export buffers.v
```

```
1 'include "header.v"
 2 //
                    -*- mode: verilog -*-
                                                                                                              2 module pa (
                                                                                                                    // chip signals
 5 // author : mike mantor
 6 // created on : sunday march 17 2002
                                                                                                                    sclk_global,
 7 // last modified by: .
                                                                                                              8
                                                                                                                     RBBM PA soft reset, //primitive assembly soft reset
 8 // last modified on:
 9 // update count : 0
                                                                                                                     CG\_PA\_pm\_enb,
10 // status : initial
                                                                                                              10
                                                                                                                     RBBM_regclk_active,
11 //-----
                                                                                                             11
                                                                                                             12
                                                                                                                   // interface to the register bus (rbbm)
13 // $id: //depot/r400/devel/parts_lib/src/gfx/pa/pa.v#34 $
                                                                                                             13
14 //
                                                                                                             14
                                                                                                                     RBBM a
                                                                                                                                       // address
15 // $change: 32279 $
                                                                                                             15
                                                                                                                     RBBM_we,
                                                                                                                                        // write enable
16 //
                                                                                                                     RBBM_wd,
                                                                                                                                         // write data
17 //
                                                                                                             17
                                                                                                                     RBBM_re,
                                                                                                                                       // read enable
                                                                                                             18 RBB_rs_in,
                                                                                                                                   // read strobe daisy chain in
18 // copyright: trade secret of ati technologies, inc.
                © copyright 2001-2002, ati technologies, inc., (unpublished)
                                                                                                             19 RBB_rs_out, // read strobe daisy chain out
                                                                                                                                   // read data daisy chain in
20 //
                                                                                                             20 RBB_rd_in,
21 //
                all rights reserved. this notice is intended as a precaution against
                                                                                                             21
                                                                                                                     RBB rd out,
                                                                                                                                     // read data daisy chain out
                 inadvertent publication and does not imply publication or any waiver
                                                                                                                    //PA_RBBM_nrtrtr, // non-real-time ready-to-receive
23 //
                of confidentiality. the year included in the foregoing notice is the
                                                                                                             23 PA_RBBM_busy,
                                                                                                                                          // busy signal reported to pa
24 //
               year of creation of the work.
                                                                                                             24 PA_a, // register address for daisy chain out
                                                                                                                                     // register we for daisy chain out
                                        Page 1 of 63
                                                                                                                                                      Page 2 of 63
                                                                        Ex. 2114 - pa.v
                                                                                                                                                                                      Ex. 2114 - pa.v
 1 PA wd,
                       // register wd for daisy chain out
                                                                                                                   VGT_PA_clip_p_dealloc, // deallocation bits
                       // register we for daisy chain out
                                                                                                                  VGT_PA_clip_p_new_vtx_vect, // primitive contains vtx that was the first vtx in a vertex vector process
      PA re,
                                                                                                                     VGT_PA_clip_p_send, // ready-to-send
       // interface to rom
                                                                                                                     PA_VGT_clip_p_rtr, // ready-to-receive
      ROM_SP0_disable,
                                                                                                                    // interface to vgt -- per state
       ROM_SP1_disable,
      ROM SP2 disable,
                                                                                                                    VGT_PA_clip_s_type, // clipper prim type. this is a sub-set of the input prim types.
       ROM_SP3_disable,
                                                                                                                     VGT_PA_clip_s_event, // event
                                                                                                             11
                                                                                                                     VGT_PA_clip_s_state, // state select
11
      // interface to vgt -- per vertex
                                                                                                             12 VGT_PA_clip_s_send, // ready-to-send
                                                                                                             13 PA_VGT_clip_s_rtr, // ready-to-receive
13 VGT_PA_clip_v_vec_size, // number of vertices in current vector
                                                                                                             14 // ---
14
      VGT_PA_clip_v_state, // state select
                                                                                                             15 // interface to the shader export 0 block
15
       VGT PA clip v send, // ready-to-send
                                                                                                             16
       PA_VGT_clip_v_rtr, // ready-to-receive
                                                                                                             17
                                                                                                                     u0_SX_PA_send,
17
                                                                                                             18
                                                                                                                    u0 SX PA data.
18
      // interface to vgt -- per primitive
                                                                                                             19 u0 PA SX req,
                                                                                                              20 u0_PA_SX_sp_id,
20 \qquad VGT\_PA\_clip\_p\_indx0, \qquad \textit{// internal vertex index 0}
                                                                                                             21 u0 PA SX offset
      VGT_PA_clip_p_indx1, // internal vertex index 1
21
                                                                                                             22
                                                                                                                    u0 PA SX aux.
       VGT_PA_clip_p_indx2, // internal vertex index 2
                                                                                                                     u0_PA_SX_last,
      VGT_PA_clip_p_edge_flags, // edge flags
24 VGT_PA_clip_p_eop, // end-of-packet for state synchronization
                                                                                                             25
                                                                                                                    // interface to the shader export 1 block
25 VGT_PA_clip_p_null_prim,
                                                                                                                                                      Page 4 of 63
                                        Page 3 of 63
                                                                        Ex. 2114 - pa.v
                                                                                                                                                                                      Ex. 2114 - pa.v
```

ATI 2114 LG v. ATI IPR2015-00326

```
ul SX PA send,
2
      ul SX PA data,
                                                                                                      2 //`include "PA_clip_pkg.v"
      ul PA SX req,
      ul_PA_SX_sp_id,
                                                                                                           ul_PA_SX_offset,
                                                                                                           // i/o definitions
      ul_PA_SX_aux,
                                                                                                           // ****************************
      ul_PA_SX_last,
      // interface to the scan converter
                                                                                                           input
                                                                                                           input RBBM_PA_soft_reset;
11
      PA_SC_p0,
                                                                                                     11
                                                                                                          input CG_PA_pm_enb;
12
      PA_SC_p1,
                                                                                                     12
                                                                                                          input RBBM_regclk_active;
13
      PA_SC_p2,
                                                                                                     13
                                                                                                           // interface to the register bus (rbbm)
14
      PA_SC_p3,
                                                                                                           input [16:2] RBBM_a;
15
      PA_SC_p4,
                                                                                                     15
                                                                                                           input RBBM we;
      PA_SC_xy0,
                                                                                                           input [31:0] RBBM_wd;
16
                                                                                                     16
17
      PA SC xv1.
                                                                                                     17
                                                                                                           input RBBM re:
18
      PA SC xv2.
                                                                                                     18
                                                                                                           input
                                                                                                                    RBB rs in:
19
      PA SC zminmax
                                                                                                     19
                                                                                                           output RBB rs out;
20
      PA SC entl
                                                                                                           input [31:0] RBB rd in;
                                                                                                     20
21
                                                                                                           output [31:0] RBB_rd_out;
      PA SC phase.
                                                                                                     21
22
      PA SC valid.
                                                                                                     22
                                                                                                           //output PA RBBM nrtrtr:
23
      PA SC v0 indx,
                                                                                                     23
                                                                                                           output PA_RBBM_busy;
24
      SC_PA_earlyfrz
                                                                                                     24
                                                                                                           output [16:2] PA_a;
25
                                                                                                           output PA_we;
                                     Page 5 of 63
                                                                                                                                           Page 6 of 63
                                                                   Ex. 2114 - pa.v
                                                                                                                                                                        Ex. 2114 - pa.v
      output [31:0] PA_wd;
                                                                                                                    VGT_PA_clip_s_event;
                                                                                                            input
2
                                                                                                      2
                                                                                                           input [2:0] VGT PA clip s state;
      output PA re;
                                                                                                                    VGT_PA_clip_s_send;
      // interface to rom
                                                                                                            input
                                                                                                                    PA_VGT_clip_s_rtr;
              ROM SP0 disable:
      input
                                                                                                           output
      input ROM SP1 disable;
                                                                                                           // interface to shader export east/0 sp0,2
      input ROM SP2 disable;
                                                                                                            input u0 SX PA send;
      input ROM_SP3_disable;
                                                                                                            input [127:0] u0_SX_PA_data;
      // interface to vgt -- per vertex
                                                                                                            output
                                                                                                                        u0 PA SX req;
      input [5:0] VGT_PA_clip_v_vec_size;
                                                                                                                        u0_PA_SX_sp_id;
      input [2:0] VGT_PA_clip_v_state;
                                                                                                            output [1:0] u0_PA_SX_offset;
10
                                                                                                     10
11
      input VGT_PA_clip_v_send;
                                                                                                     11
                                                                                                                        u0_PA_SX_aux;
12
      output PA_VGT_clip_v_rtr;
                                                                                                     12
                                                                                                                        u0_PA_SX_last;
13
      // interface to vgt -- per primitive
                                                                                                           // interface to shader export west/1 sp1,3
14
      input [5:0] VGT_PA_clip_p_indx0;
                                                                                                           input u1_SX_PA_send;
15
      input [5:0] VGT_PA_clip_p_indx1;
                                                                                                           input [127:0] u1_SX_PA_data;
      input [5:0] VGT_PA_clip_p_indx2;
                                                                                                           output
                                                                                                                        ul_PA_SX_req;
17
      input [2:0] VGT_PA_clip_p_edge_flags;
                                                                                                     17
                                                                                                                        u1_PA_SX_sp_id;
                                                                                                           output
18
      input VGT_PA_clip_p_eop;
                                                                                                     18
                                                                                                           output [1:0] u1_PA_SX_offset;
19
      input VGT_PA_clip_p_null_prim;
                                                                                                                        u1_PA_SX_aux;
                                                                                                           output
20
      input [2:0] VGT PA clip p dealloc;
                                                                                                                        ul PA SX last;
                                                                                                     20
                                                                                                           output
21
     input VGT PA clip p new vtx vect;
                                                                                                     21
                                                                                                           // interface to scan converter
22
     input VGT_PA_clip_p_send;
                                                                                                     22
                                                                                                           output [17:0] PA SC xy0;
23
      output PA_VGT_clip_p_rtr;
                                                                                                     23
                                                                                                           output [17:0] PA_SC_xy1;
24
      // interface to vgt -- per state
                                                                                                     24
                                                                                                           output [17:0] PA_SC_xy2;
      input [3:0] VGT_PA_clip_s_type;
                                                                                                           output [31:0] PA_SC_p0;
                                     Page 7 of 63
                                                                                                                                           Page 8 of 63
                                                                   Ex. 2114 - pa.v
                                                                                                                                                                        Ex. 2114 - pa.v
```

```
output [39:0] PA_SC_p1;
2
       output [31:0] PA_SC_p2;
                                                                                                                    2
       output [31:0] PA_SC_p3;
                                                                                                                               // rbbm interface wires
       output [31:0] PA_SC_p4;
                                                                                                                               wire [16:2] RBBM_a_q1;
       output [13:0] PA_SC_zminmax;
                                                                                                                                                  RBBM_we_q1;
       output [29:0] PA_SC_cntl;
                                                                                                                               wire [31:0] RBBM_wd_q1;
       output [1:0] PA_SC_phase;
                                                                                                                                                RBBM_re_q1;
       output PA_SC_valid;
                                                                                                                                                  rbiu_block_rs;
       output [1:0] PA_SC_v0_indx;
                                                                                                                               wire [31:0] rbiu_block_rd;
10
                SC_PA_earlyfrz;
                                                                                                                                            rbiu_ag_dx_clip_sp_def_sel;
11
                                                                                                                               wire [3:0] rbiu_ag_ucp0_sel;
12
       // clock gating internal wire
                                                                                                                               wire [3:0] rbiu_ag_ucp1_sel;
13
                                                                                                                               wire [3:0] rbiu_ag_ucp2_sel;
       wire
                  sclk;
14
                  sclk_reg;
                                                                                                                               wire [3:0] rbiu_ag_ucp3_sel;
       wire
15
                                                                                                                   15
                                                                                                                               wire [3:0] rbiu ag ucp4 sel;
                  sclk pa;
       wire
16
                                                                                                                    16
                                                                                                                               wire [3:0] rbiu ag ucp5 sel;
17
                                                                                                                   17
                                                                                                                               wire [3:0] rbiu_ag_gb_sel;
       wire
                  pa srst;
18
                  pa hard srst;
                                                                                                                   18
                                                                                                                               wire [3:0] rbiu_ag_pntsz_sel;
       wire
19
       wire
                  pa_soft_srst;
                                                                                                                    19
20
       wire
                  cg blk gated clk override;
                                                                                                                   20
                                                                                                                               wire
                                                                                                                                            rbiu ag cpv;
21
       wire
                  regclk_active;
                                                                                                                   21
                                                                                                                               wire [31:0] ag_rbiu_rdata;
22
       wire
                  reg_clk_en;
                                                                                                                   22
23
       wire
                  pa_clk_en;
                                                                                                                   23
                                                                                                                               wire
                                                                                                                                             rbiu_cl_dx_clip_sp_def_sel;
24
                                                                                                                   24
                                                                                                                               wire
                                                                                                                                                  rbiu_cl_status_sel;
25
                  SC_PA_earlyfrz_q;
                                                                                                                   25
                                                                                                                               wire
                                                                                                                                             rbiu_cl_cpy;
                                           Page 9 of 63
                                                                                                                                                             Page 10 of 63
                                                                             Ex. 2114 - pa.v
                                                                                                                                                                                                Ex. 2114 - pa.v
            wire [31:0] cl_rbiu_rdata;
                                                                                                                               wire
                                                                                                                                                   rbiu su point size sel;
 2
                                                                                                                    2
                                                                                                                               wire
                                                                                                                                                   rbiu_su_point_min_max_sel;
                                                                                                                                                   rbiu su line cntl sel;
            wire
                         rbiu ccg expentmd sel;
                                                                                                                               wire
                                                                                                                                                   rbiu su sc mode cntl sel;
                                                                                                                               wire
            wire
                         rbiu_ccg_cpy;
                                                                                                                               wire
                                                                                                                                                   rbiu_su_cpy;
                                                                                                                               wire [31:0] su_rbiu_rdata;
            wire [31:0] ccg_rbiu_rdata;
            wire
                               rbiu_vte_cpy;
                                                                                                                               // setup unit wires
            wire
                               rbiu_vte_xscale_sel;
                                                                                                                               // i/o for clip interface
                                                                                                                               wire [31:0] clip_su_pt_size;
10
            wire
                               rbiu_vte_xoffset_sel;
                                                                                                                    10
11
                               rbiu_vte_yscale_sel;
                                                                                                                   11
                                                                                                                               wire [17:0] clip_su_x0;
12
                               rbiu_vte_yoffset_sel;
                                                                                                                   12
                                                                                                                               wire [17:0] clip_su_x1;
13
                                rbiu_vte_zscale_sel;
                                                                                                                               wire [17:0] clip_su_x2;
14
                               rbiu_vte_zoffset_sel;
                                                                                                                   14
                                                                                                                               wire [17:0] clip_su_y0;
15
                                                                                                                               wire [17:0] clip_su_y1;
                               rbiu_vte_cntl_sel;
                                                                                                                               wire [17:0] clip_su_y2;
                               rbiu_vte_vtx_cntl_sel;
17
                               rbiu_vte_window_offset_sel;
                                                                                                                               wire [31:0] clip_su_z0;
18
                               rbiu_vte_window_offset_en_sel;
                                                                                                                               wire [31:0] clip_su_z1;
19
                               rbiu_vte_persp_corr_dis_sel;
                                                                                                                               wire [31:0] clip_su_z2;
20
            wire [31:0] vte rbiu rdata;
                                                                                                                   20
                                                                                                                               wire [31:0] clip su w0;
21
                                                                                                                   21
                                                                                                                               wire [31:0] clip su w1;
22
                                                                                                                   22
                                                                                                                               wire [31:0] clip_su_w2;
                               rbiu su imp exp sel;
            wire
23
                                                                                                                   23
            wire
                               rbiu su draw init sel:
                                                                                                                               wire [0:0] clip su ef0;
24
                                                                                                                   24
                                                                                                                               wire [0:0] clip_su_ef1;
            wire
                               rbiu su expand lw sel;
25
            wire
                               rbiu su status sel;
                                                                                                                               wire [0:0] clip_su_ef2;
                                          Page 11 of 63
                                                                                                                                                             Page 12 of 63
                                                                                                                                                                                                Ex. 2114 - pa.v
                                                                            Ex. 2114 - pa.v
```

```
wire [31:0] clip_su_i0;
 2
            wire [31:0] clip_su_i1;
                                                                                                                       // wires for common rbiu bus
            wire [31:0] clip_su_i2;
            wire [31:0] clip_su_j0;
                                                                                                                       wire rbiu_we;
           wire [31:0] clip_su_j1;
                                                                                                                           wire rbiu_re;
            wire [31:0] clip_su_j2;
                                                                                                                           wire [2:0] rbiu_waddr;
            wire [31:0] clip_su_k0;
                                                                                                                           wire [2:0] rbiu_raddr;
            wire [31:0] clip_su_k1;
                                                                                                                           wire [31:0] rbiu_wdata;
            wire [31:0] clip_su_k2;
10
            wire [10:0] clip_su_attr_indx0;
                                                                                                                      // wires for registered input
11
            wire [10:0] clip_su_attr_indx1;
                                                                                                               11
12
            wire [10:0] clip_su_attr_indx2;
                                                                                                               12
                                                                                                                      wire [46:0] VGT_PA_input_data;
13
                                                                                                               13
           wire [2:0] clip su type;
                                                                                                                      wire [46:0] VGT PA input data q;
14
           wire [2:0] clip_su_st_indx;
                                                                                                                      wire [5:0] VGT_PA_clip_v_vec_size_q;
15
                                                                                                               15
           wire [2:0] clip su dealloc slot;
                                                                                                                      wire [2:0] VGT PA clip v state q;
16
            wire [0:0] clip su null prim;
                                                                                                               16
                                                                                                                      wire VGT PA clip v send q;
17
                                                                                                               17
                                                                                                                      wire [5:0] VGT_PA_clip_p_indx0_q;
            wire [0:0] clip_su_clipped;
18
            wire [0:0] clip_su_fpov;
                                                                                                               18
                                                                                                                      wire [5:0] VGT_PA_clip_p_indx1_q;
19
            wire [0:0] clip_su_eop;
                                                                                                               19
                                                                                                                       wire [5:0] VGT_PA_clip_p_indx2_q;
                                                                                                                      wire [2:0] VGT_PA_clip_p_edge_flags_q;
20
           wire [0:0] clip su event;
                                                                                                               20
21
           wire [3:0] clip_su_event_id;
                                                                                                               21
                                                                                                                      wire VGT_PA_clip_p_eop_q;
                                                                                                                     wire VGT_PA_clip_p_null_prim_q;
22
           wire [0:0] clip_su_rts;
                                                                                                               22
23
           wire [0:0] clip_su_baryc_rts;
                                                                                                               23
                                                                                                                      wire [2:0] VGT_PA_clip_p_dealloc_q;
24
           wire [0:0] su_clip_rtr;
                                                                                                               24
                                                                                                                      wire VGT_PA_clip_p_new_vtx_vect_q;
            wire [0:0] su_clip_baryc_rtr;
                                                                                                               25
                                                                                                                       wire \qquad VGT\_PA\_clip\_p\_send\_q;
                                         Page 13 of 63
                                                                                                                                                        Page 14 of 63
                                                                          Ex. 2114 - pa.v
                                                                                                                                                                                         Ex. 2114 - pa.v
       wire [3:0] VGT_PA_clip_s_type_q;
       wire VGT_PA_clip_s_event_q;
2
                                                                                                                2
                                                                                                                       wire [17:0] PA SC xv0 q;
       wire [2:0] VGT_PA_clip_s_state_q;
                                                                                                                       wire [17:0] PA_SC_xyl_q;
       wire VGT_PA_clip_s_send_q;
                                                                                                                       wire [17:0] PA_SC_xy2_q;
5
                                                                                                                       wire [31:0] PA_SC_p0_q;
       // wires for registered output
                                                                                                                       wire [39:0] PA_SC_p1_q;
7
                                                                                                                       wire [31:0] PA_SC_p2_q;
       wire [2:0] PA_VGT_output_data;
                                                                                                                       wire [31:0] PA_SC_p3_q;
       wire [2:0] PA_VGT_output_data_q;
                                                                                                                       wire [31:0] PA_SC_p4_q;
10
       wire PA_VGT_clip_v_d;
                                                                                                               10
                                                                                                                       wire [13:0] PA_SC_zminmax_q;
11
       wire PA_VGT_clip_p_d;
                                                                                                               11
                                                                                                                       wire [29:0] PA_SC_cntl_q;
12
       wire PA_VGT_clip_s_d;
                                                                                                               12
                                                                                                                       wire [1:0] PA_SC_phase_q;
13
       wire [17:0] PA_SC_xy0_d;
                                                                                                                       wire PA_SC_valid_q;
14
       wire [17:0] PA_SC_xy1_d;
                                                                                                               14
                                                                                                                       wire [1:0] PA_SC_v0_indx_q;
15
       wire [17:0] PA_SC_xy2_d;
                                                                                                               15
                                                                                                                       wire su_busy;
       wire [31:0] PA_SC_p0_d;
17
       wire [39:0] PA_SC_p1_d;
                                                                                                               17
                                                                                                                                    u0_PA_SX_req_d;
18
       wire [31:0] PA_SC_p2_d;
                                                                                                               18
                                                                                                                                    u0_PA_SX_sp_id_d;
19
       wire [31:0] PA_SC_p3_d;
                                                                                                               19
                                                                                                                      wire [1:0] u0_PA_SX_offset_d;
20
       wire [31:0] PA SC p4 d;
                                                                                                               20
                                                                                                                      wire
                                                                                                                                    u0 PA SX aux d;
21
       wire [13:0] PA SC zminmax d;
                                                                                                                                    u0 PA SX last d;
                                                                                                               21
                                                                                                                     wire
22
       wire [29:0] PA SC cntl d;
                                                                                                               22
                                                                                                                                    ul PA SX req d;
                                                                                                                     wire
23
       wire [1:0] PA SC phase d:
                                                                                                               23
                                                                                                                                    ul PA SX sp id d;
                                                                                                                     wire
24
      wire PA_SC_valid_d;
                                                                                                                     wire [1:0] u1 PA SX offset d;
                                                                                                               24
       wire [1:0] PA SC v0 indx d;
                                                                                                               25
                                                                                                                      wire
                                                                                                                                    ul PA SX aux d;
                                         Page 15 of 63
                                                                                                                                                        Page 16 of 63
                                                                          Ex. 2114 - pa.v
                                                                                                                                                                                         Ex. 2114 - pa.v
```

1	wire ul_PA_SX_last_d;		1		
2			2	//	
3	wire u0_PA_SX_req_q;		3	// wires for pa_sxifccg	
4	wire u0_PA_SX_sp_id_q;		4	//=	
5	wire [1:0] u0_PA_SX_offset_q;		5	wire [1:0] cl_ccg_outsm_clr_orig_vertices;	
6	wire u0_PA_SX_aux_q;		6	wire cl_ccg_ccgen_to_clipcc_fifo_full;	
7	wire u0_PA_SX_last_q;		7	wire arb_ccg_xfc;	
8			8	wire ccg_ag_pos_mem_we;	
9	wire ul_PA_SX_req_q;		9	wire [5:0] ccg_ag_pntsz_mem_wraddr;	
10	wire ul_PA_SX_sp_id_q;		10	wire [5:0] ccg_ag_pos_mem_wraddr;	
11	wire [1:0] ul_PA_SX_offset_q;		11	wire [127:0] ccg_ ag_pos_pntsz_mem_wrdata;	
12	wire ul_PA_SX_aux_q; wire ul_PA_SX last q;		12	wire ccg_ag_pntsz_mem_we;	
14	wire ul_PA_SX_last_q;		14	wire [31:0] ccg_ag_pntsz_mem_wrdata; wire [17-1:0] ccg_cl_wrdata;	
15	// shader export interface/clip code generator wires		15	wire ccg_cl_write;	
16	wire [128:0] SX0_PA_input_data;		16	wire [15:0] ccg_arb_data;	
17	wire [128:0] SX0_PA_input_data_q;		17	wire [7-1:0] cl_ccg_state0;	
18	wire SX0_PA_input_data_write;		18	wire [7-1:0] cl_ccg_state1;	
19	wire [127:0] SX0_PA_input_data_wrdata;		19	wire [7-1:0] cl_ccg_state2;	
20			20	wire [7-1:0] cl_ccg_state3;	
21	wire [128:0] SX1_PA_input_data;		21	wire [7-1:0] cl_ccg_state4;	
22	wire [128:0] SX1_PA_input_data_q;		22	wire [7-1:0] cl_ccg_state5;	
23	wire SX1_PA_input_data_write;		23	wire [7-1:0] cl_ccg_state6;	
24	wire [127:0] SX1_PA_input_data_wrdata;		24	wire [7-1:0] cl_ccg_state7;	
25			25	wire [6:0] ccg_rbiu_rdata_26_downto_20;	
	Page 17 of 63	Ex. 2114 - pa.v		Page 18 of 63	Ex. 2114 - pa.v
		· 			·
1	wire [7-1:0] sxif_state0;		1	wire [29.0] cl_arb_data;	
1 2	wire [7-1:0] sxif_state0; wire [7-1:0] sxif_state1;		1 2	wire [29:0] cl_arb_data; wire vmb_cl_rei_r0vld;	
2	wire [7-1:0] sxif_state1;		2	wire vmb_cl_rei_r0vld;	
2	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2;		2 3	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld;	
2 3 4	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3;		2 3 4	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result;	
2 3 4 5 6 7	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6;		2 3 4 5 6 7	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire cl_ipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata;	
2 3 4 5 6 7 8	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5;		2 3 4 5 6 7 8	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire cl_ipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re;	
2 3 4 5 6 7 8	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7;		2 3 4 5 6 7 8	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire elipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr;	
2 3 4 5 6 7 8 9	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7;		2 3 4 5 6 7 8 9	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire elipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clp_to_ag_point_buf_re;	
2 3 4 5 6 7 8 9	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; //		2 3 4 5 6 7 8 9 10	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire elipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr;	
2 3 4 5 6 7 8 9 10 11 12	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire elipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clp_to_ag_point_buf_re;	
2 3 4 5 6 7 8	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire elipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr;	
2 3 4 5 6 7 8 9 10 11 12 13	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire elipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size;	
2 3 4 5 6 7 8 9 10 11 12	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13	wire vmb_cl_rei_r0vld; wire vmb_cl_rei_r1vld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size;	
2 3 4 5 6 7 8 9 10 11 12 13 14	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13 14	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size;	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state4; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13 14 15	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size; // // // wires for PA_ag //	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire (5:0] cl_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size; // // // wires for PA_ag // wire [2:0] ag_ve_opcode;	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // // ===============================		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire clip_to_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size; // // // wires for PA_ag // wire [2:0] ag_ve_opcode; wire [31:0] ag_ve_in_a0;	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // // // wires for pa_clipper // // wires for pa_clipper // // wire [14-1:0] clip_st_w_data; //wire clip_st_sel; wire arb_cl_xfc; wire [5:0] ag_cl_valid_bit_set; wire [3:0] ag_cl_valid_bit_set; wire [3:0] ag_cl_user_clip_indx;		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31:0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5:0] cl_ag_pntsz_mem_raddr; wire [5:0] cl_ag_point_buf_re; wire [5:0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size; // // // wires for PA_ag // wire [2:0] ag_ve_opcode; wire [31:0] ag_ve_in_a0; wire [31:0] ag_ve_in_a1;	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // // // // wires for pa_clipper // // wires for pa_clipper // //wire [14-1:0] clip_st_w_data; //wire clip_st_sel; wire arb_cl_xfc; wire [5:0] ag_cl_vertex_store_indx; wire [1:0] ag_cl_valid_bit_set; wire [3:0] ag_cl_user_clip_indx; wire ag_cl_vv_cc_test;		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire clipper_busy; wire ag_cl_pntsz_mem_blocked; wire [31-0] ag_cl_pntsz_mem_rdata; wire cl_ag_pntsz_mem_re; wire [5-0] cl_ag_pntsz_mem_raddr, wire clip_to_ag_point_buf_re; wire [5-0] clip_to_ag_point_buf_raddr; wire [31:0] ag_to_clip_point_size; // // // wires for PA_ag // wire [2-0] ag_we_opcode; wire [31:0] ag_ve_in_a0; wire [31:0] ag_ve_in_a1; wire [31:0] ag_ve_in_a2;	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire cl_rei_cl_result; wi	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; //==================================		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire cl_gontsz_memrdata; wire cl_gontsz_memredata; wire cl_gontsz_memredata; wire cl_gont_buf_re; wire [5:0] cl_gpoint_buf_re; wire [5:0] cl_gont_buf_redata; wire [31:0] agccl_ip_point_size; // // // // wires for PA_ag // // wire [2:0] agveopcode; wire [31:0] agvein_a0; wire [31:0] agvein_a2; wire [31:0] agvein_a3; wire [31:0] agvein_b0; wire [31:0] agvein_b1; wire [31:0] agvein_b1; wire [31:0] agvein_b2;	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	wire [7-1:0] sxif_state1; wire [7-1:0] sxif_state2; wire [7-1:0] sxif_state3; wire [7-1:0] sxif_state5; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state6; wire [7-1:0] sxif_state7; // ==================================		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	wire vmb_cl_rei_rlvld; wire vmb_cl_rei_rlvld; wire cl_rei_clear_result; wire cl_rei_cl_result; wi	

```
wire ag_ve_a_is_wwww;
                                                                                                                            wire [3:0] ve_veoc_vector_back_we;
2
       wire ag_ve_broadcast_x;
                                                                                                                            wire [3:0] ve_cliptemp_vector_we;
                                                                                                                            wire [127:0] ve_wdata;
       wire ag_ve_abs_a;
       wire ag_ve_abs_b;
       wire ag_ve_abs_c;
       wire ag_ve_ax_negate;
                                                                                                                           // wires for PA_ag
       wire ag_ve_ay_negate;
                                                                                                                            wire [2:0] ag_vte_opcode;
       wire ag_ve_az_negate;
       wire ag_ve_aw_negate;
                                                                                                                           wire [2:0] ag_vte_st_indx;
       wire ag_ve_bx_negate;
                                                                                                                            wire [1:0] ag_vte_vertex_store_indx;
11
       wire ag_ve_by_negate;
12
       wire ag_ve_bz_negate;
13
       wire ag_ve_bw_negate;
                                                                                                                    13
                                                                                                                           // wires for PA_vte
14
       wire ag ve cx negate;
15
                                                                                                                           wire [1:0] vte_vertex_store_indx;
                                                                                                                    15
       wire ag ve cy negate;
16
                                                                                                                    16
                                                                                                                           wire [2:0] vte opcode;
       wire ag ve cz negate;
17
                                                                                                                    17
                                                                                                                            wire [127:0] vte d;
       wire ag ve cw negate;
                                                                                                                               wire [31:0] vte_rcp_d;
18
       wire ag ve bcc flat tst;
                                                                                                                    18
19
       wire [2:0] ag_ve_out_mem_sel;
                                                                                                                    19
                                                                                                                               wire
                                                                                                                                            vte rcp rts;
20
       wire [5:0] ag ve out addr;
                                                                                                                    20
                                                                                                                               wire
                                                                                                                                            vem rep rei xfe;
21
                                                                                                                    21
                                                                                                                              wire [31:0] vcm_rcp_rei_d;
       wire [3:0] ag_ve_out_we;
22
       wire ag_ve_accum_sel;
                                                                                                                    22
23
       wire [3:0] ag_ve_pre_accum_we;
                                                                                                                    23
                                                                                                                          reg pa_sc_valid;
24
       wire [9:0] vmb_ve_tst_rtn_stat;
                                                                                                                    24
                                                                                                                           wire set_PA_RBBM_busy;
25
       wire [5:0] ve_waddr;
                                                                                                                            wire PA_RBBM_busy_d;
                                           Page 21 of 63
                                                                                                                                                              Page 22 of 63
                                                                             Ex. 2114 - pa.v
                                                                                                                                                                                                 Ex. 2114 - pa.v
       reg [3:0] PA_RBBM_busy_delay;
                                                                                                                           // register the perm clk gate override signal
       reg PA_RBBM_busy_reg;
2
                                                                                                                           ati dff in #(1) uati dff in pm en(
                                                                                                                              .clk(sclk).
       wire vcm_rcp_rei_rtr;
       wire [31:0] vmc_rei_rcp_d;
                                                                                                                              .d(CG\_PA\_pm\_enb),
       wire vmc_rei_rcp_rts;
                                                                                                                              .q(cg_blk_gated_clk_override)
                                                                                                                     6
       wire [31:0] rei_sc_r0data;
                                                                                                                     7
       wire [31:0] rei_sc_r1data;
                                                                                                                            // create registered clock enable signals based on active signals
                                                                                                                            ati_dff_in #(1) uati_dff_in_regclk_active(
10
                                                                                                                    10
                                                                                                                             .clk(sclk),
11 // pick off part of position data
                                                                                                                    11
                                                                                                                             .d(RBBM_regclk_active),
12 //=
                                                                                                                    12
                                                                                                                              .q(regclk_active)
13
       assign ccg_ag_pntsz_mem_wrdata = ccg_ag_pos_pntsz_mem_wrdata[31:0];
                                                                                                                    13
14
                                                                                                                    14
15
       assign ccg_rbiu_rdata = {5'h0, ccg_rbiu_rdata_26_downto_20, 20'h0};
                                                                                                                           // duplicate the function of the PA_SC_valid flop
16
                                                                                                                           always @(posedge sclk_pa) begin
17 //=
                                                                                                                    17
                                                                                                                           if (SC_PA_earlyfrz_q == 'h1) begin
18 // create sclk and interface to clock gating logic
                                                                                                                    18
                                                                                                                             pa_sc_valid <= PA_SC_valid_d;
19 //=
20
      //create sclk by registering sclk global
                                                                                                                    20
21
      ati master clock permanent uati master clock permanent(
                                                                                                                    21
         .clk in(sclk global),
22
                                                                                                                    22
                                                                                                                           // busy signal generation
23
         .clk out(sclk)
                                                                                                                    23
                                                                                                                           assign set_PA_RBBM_busy = VGT_PA_clip_p_send_q |
24
                                                                                                                    24
                                                                                                                                         VGT_PA_clip_s_send_q |
25
                                                                                                                    25
                                                                                                                                          VGT\_PA\_clip\_v\_send\_q \mid
                                           Page 23 of 63
                                                                                                                                                              Page 24 of 63
                                                                             Ex. 2114 - pa.v
                                                                                                                                                                                                 Ex. 2114 - pa.v
```

```
clipper busy |
 2
                       su_busy |
                                                                                                                                 //generate the sclk_reg clock tree
                       pa_sc_valid;
                                                                                                                                 ati_master_clock_gater uati_master_clock_gater_sclk_reg (
                                                                                                                                   .clk_in(sclk_global),
        always @(posedge sclk_pa) begin
                                                                                                                                   .clk(sclk),
        if (srst) begin
                                                                                                                                   .en(reg_clk_en),
          PA_RBBM_busy_delay <= 'h0;
                                                                                                                                   .pm_enb(cg_blk_gated_clk_override),
10
          PA\_RBBM\_busy\_delay \mathrel{<=} \{PA\_RBBM\_busy\_delay[2:0], set\_PA\_RBBM\_busy\};
                                                                                                                         10
11
                                                                                                                                //generate sclk_sc clock tree
12
                                                                                                                                ati_master_clock_gater uati_master_clock_gater_sc_clk (
13
                                                                                                                         13
                                                                                                                                  .clk_in(sclk_global),
14
        assign PA_RBBM_busy_d = PA_RBBM_busy_delay != 'h0;
                                                                                                                                  .clk(sclk),
15
                                                                                                                         15
                                                                                                                                  .en(pa clk en),
16
                                                                                                                                   .pm enb(cg blk gated clk override),
17
        //this enable would be intiated by RBBM_regclk_active and held high in the
                                                                                                                         17
                                                                                                                                  .clk_out(sclk_pa)
                                                                                                                         18
18
        //block as long as necessary to ensure all data could be read
                                                                                                                               ):
19
        assign reg_clk_en = regclk_active | PA_RBBM_busy;
                                                                                                                         19 //=
20
                                                                                                                         20 // instantiate common rbbm interface block
21
                                                                                                                         21 //=
       //this active signal would be a collection of request from external blocks that require
                                                                                                                               //rbbm interface register
22
        //the block clocks to be enabled along with internal busy signals that require the clocks
                                                                                                                         22
23
                                                                                                                         23
                                                                                                                                ati_rbbm_intf uati_rbbm_intf(
24
        assign\ pa\_clk\_en = !cg\_blk\_gated\_clk\_override \ |\ PA\_RBBM\_busy;
                                                                                                                         24
                                                                                                                                   .sclk_reg(sclk_reg),
25
                                                                                                                                   .rbbm_we(RBBM_we),
                                            Page 25 of 63
                                                                                                                                                                     Page 26 of 63
                                                                                Ex. 2114 - pa.v
                                                                                                                                                                                                          Ex. 2114 - pa.v
          .rbbm_re(RBBM_re),
                                                                                                                                   .d(srst),
          .rbbm_a(RBBM_a),
 2
                                                                                                                          2
                                                                                                                                   .q(pa hard srst)
          .rbbm_wd(RBBM_wd).
          .reg_we(RBBM_we_q1),
          .reg re(RBBM re q1),
                                                                                                                                 //register input soft resets
          .reg_a(RBBM_a_q1),
                                                                                                                                 ati_dff_in #(1) uati_dff_in_pa_soft_srst(
          .reg_wd(RBBM_wd_q1),
                                                                                                                                   .clk(sclk).
                                                                                                                                   .d(RBBM_PA_soft_reset),
          .pipe_we(PA_we),
          .pipe_re(PA_re),
                                                                                                                                   .q(pa_soft_srst)
          .pipe_a(PA_a),
10
                                                                                                                         10
11
          .pipe_wd(PA_wd),
                                                                                                                         11
12
          .rbbm_rs_in(RBB_rs_in),
                                                                                                                         12
                                                                                                                                //use this in the block for the srst everywhere except
13
          .rbbm_rd_in(RBB_rd_in),
                                                                                                                                //state storage that can only get reset by a hard reset
14
          .block_rs(rbiu_block_rs),
                                                                                                                                 assign pa_srst = pa_soft_srst | pa_hard_srst;
15
          .block_rd(rbiu_block_rd),
          .rbbm_rs_out(RBB_rs_out),
17
          .rbbm_rd_out(RBB_rd_out)
                                                                                                                         17 // register outputs
18
                                                                                                                         18 //=
19
                                                                                                                               ati_dff_out #(3) uVGT_PA_out_intf(
20
                                                                                                                         20
                                                                                                                                  .clk(sclk pa),
21
    // create hard and soft reset signal
                                                                                                                         21
                                                                                                                                  .d(PA VGT output data),
22
                                                                                                                         22
                                                                                                                                   .q(PA VGT output data q)
23
                                                                                                                         23
       //register input reset
24
       ati\_dff\_in\ \#(1)\ uati\_dff\_in\_pa\_hard\_srst(
                                                                                                                         24
          clk(sclk)
                                                                                                                                ati_dff_en_out #(32) uPA_SC_out0(
                                            Page 27 of 63
                                                                                                                                                                     Page 28 of 63
                                                                                Ex. 2114 - pa.v
                                                                                                                                                                                                         Ex. 2114 - pa.v
```

```
.clk(sclk pa),
2
         .en(SC\_PA\_earlyfrz\_q),
                                                                                                                    ati_dff_en_out #(32) uPA_SC_out4(
         .d(PA\_SC\_p0\_d),
          .q(PA_SC_p0_q)
                                                                                                                      .clk(sclk_pa),
                                                                                                                       .en(SC_PA_earlyfrz_q),
                                                                                                                       .d(PA\_SC\_p4\_d),
       ati_dff_en_out #(40) uPA_SC_out1(
                                                                                                                       .q(PA_SC_p4_q)
         .en(SC_PA_earlyfrz_q),
         .d(PA_SC_p1_d),
                                                                                                                    ati_dff_en_out #(54) uPA_SC_out5(
11
         .q(PA\_SC\_p1\_q)
                                                                                                              11
                                                                                                                     .clk(sclk_pa),
12
                                                                                                                    .en(SC_PA_earlyfrz_q),
13
                                                                                                                      .d({PA_SC_xy2_d,PA_SC_xy1_d,PA_SC_xy0_d}),
14
       ati_dff_en_out #(32) uPA_SC_out2(
                                                                                                                      .q(\{PA\_SC\_xy2\_q,PA\_SC\_xy1\_q,PA\_SC\_xy0\_q\})
15
                                                                                                              15
         .clk(sclk pa),
         .en(SC_PA_earlyfrz_q),
                                                                                                              16
16
17
         .d(PA_SC_p2_d),
                                                                                                              17
                                                                                                                    ati_dff_en_out #(49) uPA_SC_out6(
                                                                                                             18
18
         .q(PA\_SC\_p2\_q)
                                                                                                                      .clk(sclk pa).
19
                                                                                                              19
                                                                                                                      .en(SC_PA_earlyfrz_q),
20
                                                                                                                   .d({PA_SC_v0_indx_d,PA_SC_valid_d,PA_SC_cntl_d,PA_SC_phase_d,PA_SC_zminmax_d})
21
      ati_dff_en_out #(32) uPA_SC_out3(
22
         .clk(sclk pa),
                                                                                                                   .q({PA_SC_v0_indx_q,PA_SC_valid_q,PA_SC_cntl_q,PA_SC_phase_q,PA_SC_zminmax_q})
23
         .en(SC_PA_earlyfrz_q),
24
         .d(PA_SC_p3_d),
                                                                                                              26
         .q(PA_SC_p3_q)
                                                                                                              27
                                        Page 29 of 63
                                                                                                                                                      Page 30 of 63
                                                                         Ex. 2114 - pa.v
                                                                                                                                                                                       Ex. 2114 - pa.v
                                                                                                                                     ul_PA_SX_sp_id_d,
      // u0_PA_SX
                                                                                                                                     ul_PA_SX_offset_d,
2
                                                                                                                                     ul_PA_SX_aux_d,
                                                                                                                                     ul_PA_SX_last_d}),
      ati_dff_out #(6) uPA_SX0_data_out (
                                                                                                                       .q({ul_PA_SX_req_q,
         .clk(sclk pa),
                                                                                                                        ul_PA_SX_sp_id_q,
         .d(\{u0\_PA\_SX\_req\_d,
                                                                                                                         ul_PA_SX_offset_q,
                      u0_PA_SX_sp_id_d,
                                                                                                                         ul_PA_SX_aux_q,
                       u0_PA_SX_offset_d,
                                                                                                                         u1\_PA\_SX\_last\_q\})
10
                       u0_PA_SX_aux_d,
                                                                                                              10
11
                       u0_PA_SX_last_d}),
                                                                                                             11
12
         .q(\{u0\_PA\_SX\_req\_q,
                                                                                                             12
          u0_PA_SX_sp_id_q,
                                                                                                                    // PA_RBBM_busy
          u0_PA_SX_offset_q,
15
           u0_PA_SX_aux_q,
                                                                                                                    ati_dff_out #(1) uPA_RBBM_busy (
           u0\_PA\_SX\_last\_q\})
                                                                                                                     .clk(sclk_pa),
17
                                                                                                                      .d(PA_RBBM_busy_d),
18
                                                                                                              18
                                                                                                                      .q(PA_RBBM_busy)
19
20
      // ul_PA_SX
                                                                                                             20
21
                                                                                                             21
22
                                                                                                             22
                                                                                                                    // register inputs
23
     ati\_dff\_out\,\#(6)\,uPA\_SX1\_data\_out\,(
                                                                                                             23
24
                                                                                                             24
                                                                                                                    ati_dff_in #(47) uVGT_PA_in_inft(
        .clk(sclk pa),
25
        .d({u1_PA_SX_req_d,
                                                                                                                      .clk(sclk_pa),
                                        Page 31 of 63
                                                                                                                                                      Page 32 of 63
                                                                                                                                                                                       Ex. 2114 - pa.v
                                                                         Ex. 2114 - pa.v
```

```
assign ul PA SX last
                                                                                                                                                    = ul PA SX last a:
         .d(VGT PA input data),
2
                                                                                                                 assign PA_SC_v0_indx = PA_SC_v0_indx_q;
         .q(VGT\_PA\_input\_data\_q)
3
                                                                                                                  assign PA_SC_valid = PA_SC_valid_q;
4
                                                                                                                  assign PA_SC_cntl = PA_SC_cntl_q;
5
      ati_dff_in #(129) uu0_SX_PA_data_inft(
                                                                                                                  assign PA_SC_phase = PA_SC_phase_q;
        .clk(sclk_pa),
                                                                                                                  assign PA_SC_zminmax = PA_SC_zminmax_q;
        .d(SX0_PA_input_data),
                                                                                                                  assign PA_SC_xy0 = PA_SC_xy0_q;
                                                                                                                  assign PA_SC_xyl = PA_SC_xyl_q;
         .q(SX0\_PA\_input\_data\_q)
                                                                                                                  assign PA_SC_xy2 = PA_SC_xy2_q;
10
                                                                                                                 assign PA_SC_p0 = PA_SC_p0_q;
11
      ati_dff_in #(129) uu1_SX_PA_data_inft(
                                                                                                                 assign PA_SC_p1
                                                                                                                                       = PA_SC_p1_q;
12
        .clk(sclk_pa),
                                                                                                                 assign PA_SC_p2
                                                                                                                                       = PA_SC_p2_q;
13
         .d(SX1_PA_input_data),
                                                                                                           13
                                                                                                                 assign PA_SC_p3
                                                                                                                                       = PA_SC_p3_q;
14
         .q(SX1_PA_input_data_q)
                                                                                                                  assign PA_SC_p4
                                                                                                                                       = PA_SC_p4_q;
15
                                                                                                           15
      );
16
                                                                                                           16
                                                                                                                  assign {VGT PA clip v vec size q, // [46:41]
17
                                               = u0 PA SX req q;
                                                                                                           17
           assign u0 PA SX req
                                                                                                                      VGT PA clip v state q, // [40:38]
18
           assign u0 PA SX sp id
                                         = u0 PA_SX_sp_id_q;
                                                                                                           18
                                                                                                                      VGT_PA_clip_v_send_q, // [37]
19
           assign u0 PA SX offset
                                         = u0 PA SX offset q;
                                                                                                           19
                                                                                                                      VGT_PA_clip_p_indx0_q, // [36:31]
                                                                                                                      VGT_PA_clip_p_indx1_q, // [30:25]
20
           assign u0 PA SX aux
                                              = u0 PA SX aux q;
                                                                                                           20
21
           assign u0 PA SX last
                                         = u0 PA SX last q;
                                                                                                           21
                                                                                                                      VGT_PA_clip_p_indx2_q, // [24:19]
22
           assign u1 PA SX req
                                              = ul PA SX req q;
                                                                                                           22
                                                                                                                      VGT_PA_clip_p_edge_flags_q, // [18:16]
23
           assign u1_PA_SX_sp_id
                                        = u1_PA_SX_sp_id_q;
                                                                                                           23
                                                                                                                      VGT_PA_clip_p_eop_q,
24
           assign u1_PA_SX_offset
                                         = u1_PA_SX_offset_q;
                                                                                                           24
                                                                                                                      VGT_PA_clip_p_null_prim_q, // [14]
25
           assign u1_PA_SX_aux
                                              = u1_PA_SX_aux_q;
                                                                                                           25
                                                                                                                       VGT\_PA\_clip\_p\_dealloc\_q, \qquad /\!/ \left[ 13:11 \right]
                                       Page 33 of 63
                                                                                                                                                   Page 34 of 63
                                                                       Ex. 2114 - pa.v
                                                                                                                                                                                   Ex. 2114 - pa.v
           VGT_PA_clip_p_new_vtx_vect_q, // [10]
                                                                                                                  assign PA_VGT_clip_v_rtr = PA_VGT_output_data_q[2];
 2
           VGT PA clip p send q, //[9]
                                                                                                                  assign PA VGT clip p rtr = PA VGT output data q[1];
           VGT_PA_clip_s_event_q, // [8]
                                                                                                                   assign\ PA\_VGT\_clip\_s\_rtr \qquad = PA\_VGT\_output\_data\_q[0];
           VGT_PA_clip_s_type_q, // [7:4]
           VGT_PA_clip_s_state_q, // [3:1]
                                                                                                                  assign PA_VGT_output_data = {PA_VGT_clip_v_d,
           VGT\_PA\_clip\_s\_send\_q\} \hspace{1.5cm} /\!/ \left[0\right]
                                                                                                                                    PA VGT clip p d,
                         = VGT_PA_input_data_q;
                                                                                                                                    PA_VGT_clip_s_d};
       assign VGT_PA_input_data = {VGT_PA_clip_v_vec_size, // [46:41]
                                                                                                                   assign\ SX0\_PA\_input\_data = \{u0\_SX\_PA\_send,
10
                    VGT_PA_clip_v_state, // [40:38]
                                                                                                           10
                                                                                                                                 u0\_SX\_PA\_data\};
11
                     VGT_PA_clip_v_send, // [37]
                                                                                                           11
12
                    VGT_PA_clip_p_indx0, // [36:31]
                                                                                                           12
                                                                                                                  assign {SX0_PA_input_data_write,
                    VGT_PA_clip_p_indx1, // [30:25]
                                                                                                           13
                                                                                                                      SX0_PA_input_data_wrdata} = SX0_PA_input_data_q;
                    VGT_PA_clip_p_indx2, // [24:19]
                                                                                                           14
                     VGT_PA_clip_p_edge_flags, // [18:16]
                                                                                                                  assign SX1_PA_input_data = {u1_SX_PA_send,
                    VGT_PA_clip_p_eop, // [15]
                                                                                                                                 u1\_SX\_PA\_data\};
17
                     VGT_PA_clip_p_null_prim, //[14]
                                                                                                           17
18
                     VGT_PA_clip_p_dealloc, // [13:11]
                                                                                                           18
                                                                                                                  assign {SX1_PA_input_data_write,
                     VGT_PA_clip_p_new_vtx_vect,// [10]
                                                                                                                       SX1\_PA\_input\_data\_wrdata\} = SX1\_PA\_input\_data\_q;
20
                     VGT PA clip p send, //[9]
                                                                                                           20
21
                     VGT PA clip s event, //[8]
                                                                                                           21
22
                     VGT_PA_clip_s_type, // [7:4]
                                                                                                           22
                                                                                                                 //register pipe freeze signal before using it
23
                     VGT_PA_clip_s_state, // [3:1]
                                                                                                           23
                                                                                                                 ati dff in #(1) uati_dff_in_earlyfrz(
24
                     VGT_PA_clip_s_send}; // [0]
                                                                                                           24
                                                                                                                   clk(sclk_na)
25
                                                                                                                   .d(SC_PA_earlyfrz),
                                       Page 35 of 63
                                                                                                                                                   Page 36 of 63
                                                                       Ex. 2114 - pa.v
                                                                                                                                                                                   Ex. 2114 - pa.v
```

```
.q(SC PA earlyfrz q)
                                                                                                                                           .rbiu_ag_ucp0_sel(rbiu_ag_ucp0_sel),
2
                                                                                                                                           .rbiu\_ag\_ucp1\_sel(rbiu\_ag\_ucp1\_sel),
                                                                                                                                           .rbiu_ag_ucp2_sel(rbiu_ag_ucp2_sel),
 4 // rbbm interface
                                                                                                                                           .rbiu_ag_ucp3_sel(rbiu_ag_ucp3_sel),
      pa_rbiu upa_rbiu (
                                                                                                                                           .rbiu_ag_ucp4_sel(rbiu_ag_ucp4_sel),
            // chip signals
                                                                                                                                           .rbiu\_ag\_ucp5\_sel(rbiu\_ag\_ucp5\_sel),
             .CG_PA_sclk_reg(sclk_reg),
                                                                                                                                           .rbiu_ag_gb_sel(rbiu_ag_gb_sel),
             // interface to the global register bus (rbbm)
                                                                                                                                           .rbiu_ag_pntsz_sel(rbiu_ag_pntsz_sel),
             .RBBM_a_ql(RBBM_a_ql),
10
                                                                                                                              10
             .RBBM\_we\_ql(RBBM\_we\_ql),
                                                                                                                                           .cl_rbiu_rdata(cl_rbiu_rdata),
11
             .RBBM\_wd\_q1(RBBM\_wd\_q1),
                                                              // write data
                                                                                                                                           .rbiu_cl_cpy(rbiu_cl_cpy),
12
             .RBBM_re_q1(RBBM_re_q1),
                                                               // read enable
                                                                                                                              12
                                                                                                                                           .rbiu\_cl\_dx\_clip\_sp\_def\_sel(rbiu\_cl\_dx\_clip\_sp\_def\_sel),\\
13
            // rbbm read data daisy chain
                                                                                                                              13
                                                                                                                                           .rbiu_cl_status_sel(rbiu_cl_status_sel),
14
             .rbiu_block_rs(rbiu_block_rs),
                                                          // read strobe daisy chain out
                                                                                                                                           // interface to ccg
15
             .rbiu block rd(rbiu block rd),
                                                        // read data daisy chain in
                                                                                                                              15
                                                                                                                                           .ccg rbiu rdata(ccg rbiu rdata).
16
        // common rbiu interface to vte,su,ccg,cl,ag
                                                                                                                              16
                                                                                                                                           .rbiu ccg cpv(rbiu ccg cpv),
17
                                                                                                                              17
        .rbiu we(rbiu we).
                                                                                                                                           .rbiu_ccg_expcntmd_sel(rbiu_ccg_expcntmd_sel),
18
        .rbiu re(rbiu re).
                                                                                                                              18
                                                                                                                                           // interface to vte
19
        rbin waddr(rbin waddr)
                                                                                                                              19
                                                                                                                                           .vte rbiu rdata(vte rbiu rdata),
20
        rbiu raddr(rbiu raddr)
                                                                                                                              20
                                                                                                                                           .rbiu vte cpy(rbiu vte cpy),
21
        .rbiu wdata(rbiu wdata).
                                                                                                                             21
                                                                                                                                           .rbiu vte xscale sel(rbiu vte xscale sel),
22
            // interface to ag
                                                                                                                             22
                                                                                                                                           .rbiu vte xoffset sel(rbiu vte xoffset sel).
23
             .ag_rbiu_rdata(ag_rbiu_rdata),
                                                                                                                             23
                                                                                                                                           .rbiu_vte_yscale_sel(rbiu_vte_yscale_sel),
24
             .rbiu_ag_cpy(rbiu_ag_cpy),
                                                                                                                             24
                                                                                                                                           .rbiu_vte_yoffset_sel(rbiu_vte_yoffset_sel),
25
              .rbiu_ag_dx_clip_sp_def_sel(rbiu_ag_dx_clip_sp_def_sel),
                                                                                                                             25
                                                                                                                                           .rbiu_vte_zscale_sel(rbiu_vte_zscale_sel),
                                              Page 37 of 63
                                                                                                                                                                           Page 38 of 63
                                                                                   Ex. 2114 - pa.v
                                                                                                                                                                                                                 Ex. 2114 - pa.v
             .rbiu_vte_zoffset_sel(rbiu_vte_zoffset_sel),
                                                                                                                                     .rbiu_ag_dx_clip_sp_def_sel (rbiu_ag_dx_clip_sp_def_sel),
             .rbiu vte cntl sel(rbiu vte cntl sel),
                                                                                                                                     .rbiu_ag_ucp0_sel
                                                                                                                                                                 (rbiu ag ucp0 sel),
             .rbiu vte vtx cntl sel(rbiu vte vtx cntl sel),
                                                                                                                                     .rbiu_ag_ucp1_sel
                                                                                                                                                                 (rbiu_ag_ucpl_sel),
             .rbiu vte window offset sel(rbiu vte window offset sel),
                                                                                                                                     .rbiu_ag_ucp2_sel
                                                                                                                                                                 (rbiu_ag_ucp2_sel),
             .rbiu_vte_window_offset_en_sel(rbiu_vte_window_offset_en_sel),
                                                                                                                                     .rbiu_ag_ucp3_sel
                                                                                                                                                                 (rbiu_ag_ucp3_sel),
             .rbiu_vte_persp_corr_dis_sel(rbiu_vte_persp_corr_dis_sel),
                                                                                                                                     .rbiu_ag_ucp4_sel
                                                                                                                                                                 (rbiu_ag_ucp4_sel),
             // interface to su
                                                                                                                                     .rbiu_ag_ucp5_sel
                                                                                                                                                                 (rbiu_ag_ucp5_sel),
             .su rbiu rdata(su rbiu rdata),
                                                                                                                                     .rbiu_ag_gb_sel
                                                                                                                                                                (rbiu_ag_gb_sel),
             .rbiu_su_cpy(rbiu_su_cpy),
                                                                                                                                     .rbiu_ag_pntsz_sel
                                                                                                                                                                 (rbiu_ag_pntsz_sel),
10
             .rbiu_su_expand_lw_sel(rbiu_su_expand_lw_sel),
                                                                                                                              10
11
             .rbiu_su_imp_exp_sel(rbiu_su_imp_exp_sel),
                                                                                                                             11
                                                                                                                                     .rbiu_we
                                                                                                                                                              (rbiu_we),
12
             .rbiu_su_draw_init_sel(rbiu_su_draw_init_sel),
                                                                                                                             12
                                                                                                                                     .rbiu_wa
                                                                                                                                                              (rbiu_waddr),
13
                                                                                                                                                              (rbiu_wdata),
             .rbiu_su_status_sel(rbiu_su_status_sel),
                                                                                                                                                              (rbiu_su_cpy),
14
             .rbiu_su_point_size_sel(rbiu_su_point_size_sel),
                                                                                                                                     .rbiu_cpy
15
             .rbiu_su_point_min_max_sel(rbiu_su_point_min_max_sel),
                                                                                                                                     .rbiu_re
                                                                                                                                                             (rbiu_re),
16
             .rbiu\_su\_line\_cntl\_sel(rbiu\_su\_line\_cntl\_sel),
                                                                                                                                     .ag_rbiu_rd
                                                                                                                                                              (ag_rbiu_rdata),
17
             .rbiu\_su\_sc\_mode\_cntl\_sel(rbiu\_su\_sc\_mode\_cntl\_sel)
                                                                                                                              17
18
                                                                                                                              18
                                                                                                                                     .ccg to arb data
                                                                                                                                                                 (ccg_arb_data),
19
                                                                                                                              19
                                                                                                                                     arb ccg xfc
                                                                                                                                                               (arb ccg xfc),
20
                                                                                                                             20
     pa ag
21
                                                                                                                             21
                                                                                                                                                                (cl arb ve valid)
                                                                                                                                     .clip ve valid
     upa ag
22
                                                                                                                             22
                                                                                                                                                                (cl arb data).
                                                                                                                                     .clip to arb data
      (
23
                             (sclk_pa)
                                                                                                                             23
       .sclk
                                                                                                                                                               (arb cl xfc),
                                                                                                                                     .arb clip xfc
24
                                                                                                                             24
                             (srst),
25
                                                                                                                                     .clip_to_ag_point_buf_re
                                                                                                                                                                    (clip_to_ag_point_buf_re)
                                              Page 39 of 63
                                                                                                                                                                           Page 40 of 63
                                                                                   Ex. 2114 - pa.v
                                                                                                                                                                                                                 Ex. 2114 - pa.v
```

 clip 	p_to_ag_point_buf_raddr	(clip_to_ag_point_buf_raddr),		1			
	to_clip_point_size	(ag_to_clip_point_size),		2	.ag_vte_opcode	(ag_vte_opcode),	
3				3	.ag_vte_st_indx	(ag_vte_st_indx),	
4 .pos_	s_pntsz_ag_mem_data	(ccg_ag_pos_pntsz_mem_wrdata),		4	.ag_vte_vertex_store_indx		
5 .pos_	s_mem_waddr	(ccg_ag_pos_mem_wraddr),		5			
6 .pnts	tsz_mem_waddr	(ccg_ag_pntsz_mem_wraddr),		6	.ag_ve_opcode	(ag_ve_opcode),	
7 .pos_	s_mem_we	(ccg_ag_pos_mem_we),		7	.ag_ve_in_a0	(ag_ve_in_a0),	
8 .pnts	tsz_mem_we	(ccg_ag_pntsz_mem_we),		8	.ag_ve_in_al	(ag_ve_in_al),	
9				9	.ag_ve_in_a2	(ag_ve_in_a2),	
10 .inv_	_ret_sc_data (({rei_sc_rldata,rei_sc_r0data}),		10	.ag_ve_in_a3	(ag_ve_in_a3),	
11				11	.ag_ve_in_b0	(ag_ve_in_b0),	
12 .ve_0	_cliptemp_vector_we	(ve_cliptemp_vector_we),		12	.ag_ve_in_b1	(ag_ve_in_bl),	
13 .ve_v	_veoc_vector_back_we	(ve_veoc_vector_back_we),		13	.ag_ve_in_b2	(ag_ve_in_b2),	
14 .ve_v	_waddr (v	ve_waddr),		14	.ag_ve_in_b3	(ag_ve_in_b3),	
15 .ve_v	_wdata (v	re_wdata),		15	.ag_ve_a_is_wwww	(ag_ve_a_is_wwww),	
16				16	.ag_ve_broadcast_x	(ag_ve_broadcast_x),	
17 .agve	ve_dly_valid_op	(),		17	.ag_ve_abs_a	(ag_ve_abs_a),	
18 .agve	ve_dly_vertex_store_indx	(ag_cl_vertex_store_indx),		18	.ag_ve_abs_b	(ag_ve_abs_b),	
19 .agve	ve_dly_valid_bit_set	(ag_cl_valid_bit_set),		19	.ag_ve_abs_c	(ag_ve_abs_c),	
20 .agve	ve_dly_user_clip_indx	(ag_cl_user_clip_indx),		20	.ag_ve_ax_negate	(ag_ve_ax_negate),	
21 .agve	ve_dly_vv_cc_test	(ag_cl_vv_cc_test),		21	.ag_ve_ay_negate	(ag_ve_ay_negate),	
22 .agve	ve_dly_ucp_cc_test	(ag_cl_ucp_cc_test),		22	.ag_ve_az_negate	(ag_ve_az_negate),	
23 .agve	ve_dly_bcc_cc_test	(ag_cl_bcc_cc_test),		23	.ag_ve_aw_negate	(ag_ve_aw_negate),	
24 .agve	ve_dly_ps_ucp_cc_test	(ag_cl_ps_ucp_cc_test),		24	.ag_ve_bx_negate	(ag_ve_bx_negate),	
25 .agve	ve_dly_ps_engh_test	(ag_cl_ps_engh_test),		25	.ag_ve_by_negate	(ag_ve_by_negate),	
		Page 41 of 63				Page 42 of 63	
			Ex. 2114 - pa.v				Ex. 2114 - pa.v
1 .ag v	ve bz negate	(ag ve bz negate).		1	.iag ve in b3	(ag ve in b3),	
	_ve_bz_negate _ve_bw_negate	(ag_ve_bz_negate), (ag_ve_bw_negate),		1 2	.iag_ve_in_b3	(ag_ve_in_b3), (ag_ve_a_is_www),	
2 .ag_	_ve_bw_negate	(ag_ve_bw_negate),			.iag_ve_a_is_wwww	(ag_ve_a_is_wwww),	
2 .ag_v 3 .ag_v				2			
2 .ag_v 3 .ag_v 4 .ag_v	_ve_bw_negate _ve_cx_negate	(ag_ve_bw_negate), (ag_ve_cx_negate),		2 3	.iag_ve_a_is_wwww .iag_ve_broadcast_x	(ag_ve_a_is_wwww), (ag_ve_broadcast_x),	
2 .ag_\cdot 3 .ag_\cdot 4 .ag_\cdot 5 .ag_\cdot	_ve_bw_negate _ve_cx_negate _ve_cy_negate	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate),		2 3 4	.iag_ve_a_is_wwww .iag_ve_broadcast_x .iag_ve_abs_a	(ag_ve_a_is_wwww), (ag_ve_broadcast_x), (ag_ve_abs_a),	
2 .ag_\cdot 3 .ag_\cdot 4 .ag_\cdot 5 .ag_\cdot 6 .ag_\cdot	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate),		2 3 4 5	.iag_ve_a_is_wwww .iag_ve_broadcast_x .iag_ve_abs_a .iag_ve_abs_b	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_'	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cz_negate	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cw_negate),		2 3 4 5 6	.iag_ve_a_is_wwww .iag_ve_broadcast_x .iag_ve_abs_a .iag_ve_abs_b .iag_ve_abs_c	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_'	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cz_negate ve_cw_negate ve_bcc_flat_tst vve_out_mem_sel	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst),		2 3 4 5 6 7	iag_ve_a_is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_as_c	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_abs_c),	
2 .ag_1 3 .ag_1 4 .ag_1 5 .ag_1 7 .ag_1 8 .ag_1 9 .ag_1	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cw_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel),		2 3 4 5 6 7 8	iag_ve_a_is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_as_c), (ag_ve_av_negate), (ag_ve_av_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_'	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cw_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr),		2 3 4 5 6 7 8	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate iag_ve_az_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_abs_c), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_'	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cw_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr ve_out_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we),		2 3 4 5 6 7 8 9	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate iag_ve_az_negate iag_ve_az_negate iag_ve_aw_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_abs_c), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_'	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cz_negate ve_bcc_flat_ist ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_st), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel),		2 3 4 5 6 7 8 9	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_abs_c), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_'	ve_bw_negate ve_cx_negate ve_cy_negate ve_cz_negate ve_cz_negate ve_bcc_flat_ist ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_st), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel),		2 3 4 5 6 7 8 9 10	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_as_e iag_ve_as_negate iag_ve_as_negate iag_ve_as_negate iag_ve_as_negate iag_ve_bs_negate iag_ve_bs_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_as_ec), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_bv_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13);	ve_bw_negate ve_ex_negate ve_ex_negate ve_ez_negate ve_ex_negate ve_ew_negate ve_bec_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_st), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel),		2 3 4 5 6 7 8 9 10 11 12 13	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_as_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_as_ec), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_bv_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13);	ve_bw_negate ve_ex_negate ve_ex_negate ve_ez_negate ve_ex_negate ve_ex_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_we ve_accum_sel ve_eaccum_sel ve_re_accum_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_st), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel),		2 3 4 5 6 7 8 9 10 11 12 13	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 pa_cl_ 16 upa_c	ve_bw_negate ve_ex_negate ve_ex_negate ve_ez_negate ve_ex_negate ve_ex_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_we ve_accum_sel ve_eaccum_sel ve_re_accum_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_st), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel),		2 3 4 5 6 7 8 9 10 11 12 13 14	iag_ve_a is_www iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate iag_ve_by_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_as_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_bs_ec), (ag_ve_cs_ec), (ag_ve_cs_ec),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 pa_cl_ 16 upa_c	ve_bw_negate ve_ex_negate ve_ex_negate ve_cz_negate ve_cz_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cx_negate), (ag_ve_bcc_flat_st), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel),		2 3 4 5 6 7 8 9 10 11 12 13 14 15	iag_ve_a_is_www iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_cx_negate	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_ax_negate), (ag_ve_ax_negate), (ag_ve_ax_negate), (ag_ve_ax_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_ve_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 pa_cl_ 16 upa_c 17 (18 .iag_'	ve_bw_negate ve_ex_negate ve_ex_negate ve_cz_negate ve_cz_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_we ve_accum_sel ve_pre_accum_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cz_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel), (ag_ve_pre_accum_we)		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iag_ve_a is_www iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ay_negate iag_ve_az_negate iag_ve_az_negate iag_ve_by_negate	(ag_ve_a_is_www), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_abs_c), (ag_ve_ax_negate), (ag_ve_ax_negate), (ag_ve_ax_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_bx_negate), (ag_ve_cx_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cz_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 pa_cl 16 upa_c 17 (18 .iag 19 .iag_' 20 .iag_'	ve_bw_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we l_ve cl_ve ve_uve_out_mem_sel ve_pre_accum_we	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel), (ag_ve_pre_accum_we) (ag_ve_pre_accum_we)		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	iag_ve_a is_www iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_by_negate iag_ve_by_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cy_negate iag_ve_cx_negate iag_ve_bc_flat_tst iag_ve_out_mem_sel	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_as_negate), (ag_ve_av_negate), (ag_ve_av_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_bv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_ve_negate), (ag_ve_ve_ve_negate), (ag_ve_ve_ve_negate), (ag_ve_ve_ve_ve_ve_ve_ve_ve_ve_ve_ve_ve_ve_	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 pa_cl_ 16 upa_c 17 (18 .iag_ 19 .iag_ 19 .iag_ 20 .iag_ 21 .iag_'	ve_bw_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we I_ve cl_ve ve_uve_in_a0 (_ve_in_a1 (_ve_in_a2 (_ve_in_a1)	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel), (ag_ve_pre_accum_we) (ag_ve_pre_accum_we)		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_as_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_ay_negate iag_ve_by_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cy_negate iag_ve_cy_negate iag_ve_cy_negate iag_ve_bcc_flat_tst iag_ve_out_mem_sel iag_ve_out_addr	(ag_ve_a_is_www), (ag_ve_abs_a), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_as_e), (ag_ve_as_negate), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_cz_negate), (ag_ve_cz_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_ve_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 9 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 pa_cl_ 16 upa_c 17 (18 .iag_ 19 .iag_ 19 .iag_ 20 .iag_ 21 .iag_ 22 .iag_'	ve_bw_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_bcc_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_accum_we l_ve cl_ve cl_ve _ve_in_a0 (_ve_in_a1 (_ve_in_a2 (_ve_in_a3)	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cw_negate), (ag_ve_bcc_flat_tst), (ag_ve_bcc_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_we), (ag_ve_accum_sel), (ag_ve_pre_accum_we) (ag_ve_pre_accum_we)		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_abs_c iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_by_negate iag_ve_by_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_cx_negate iag_ve_out_mem_sel iag_ve_out_mem_sel iag_ve_out_addr iag_ve_out_we	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_as_e), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_ov_negate),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 15 .pa_cl 16 .upa_c 17 (18 .iag_ 19 .iag_ 20 .iag_ 21 .iag_ 22 .iag_ 23 .iag_ 23 .iag_ 3 .ag_'	ve_bw_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_bce_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we I_ve cl_ve _ve_in_a0 ((_ve_in_a1 ((_ve_in_a2 ((_ve_in_b0 ((_ve_in_b0 ((_ve_in_a3 ((_ve_in_b0 ((_v	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cw_negate), (ag_ve_bec_flat_tst), (ag_ve_bec_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_sel), (ag_ve_pre_accum_we) (ag_ve_pre_accum_we) (ag_ve_in_a0), (ag_ve_in_a1), (ag_ve_in_a2), (ag_ve_in_b0),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	iag ve a is wwww iag ve broadcast x iag ve abs a iag ve abs b iag ve abs c iag ve abs c iag ve as c iag ve as c iag ve as c iag ve as negate iag ve as negate iag ve as negate iag ve as negate iag ve bas negate iag ve bs negate iag ve bs negate iag ve bs negate iag ve bs negate iag ve c negate iag ve c c negate iag ve cy negate iag ve cy negate iag ve co negate iag ve cou negate iag ve out mem sel iag ve out mem sel iag ve out we iag ve accum sel	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_as_e), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_cz_negate), (ag_ve_cz_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_out_mer_sel), (ag_ve_out_mer_sel), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_out_we),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 .ag_' 15 .pa_cl_ 16 .upa_c 17 (18 .iag_ 19 .iag_ 20 .iag_ 21 .iag_ 22 .iag_ 23 .iag_ 24 .iag_ 24 .iag_	ve_bw_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_bce_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we I_ve cl_ve _ve_in_a0	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cw_negate), (ag_ve_bec_flat_tst), (ag_ve_bec_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_sel), (ag_ve_pre_accum_we) (ag_ve_pre_accum_we) (ag_ve_in_a0), (ag_ve_in_a1), (ag_ve_in_a2), (ag_ve_in_b0), (ag_ve_in_b0), (ag_ve_in_b1),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	iag_ve_a is_wwww iag_ve_broadcast_x iag_ve_abs_a iag_ve_abs_b iag_ve_abs_c iag_ve_abs_c iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_ax_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_bx_negate iag_ve_cx_negate	(ag_ve_a_is_www), (ag_ve_abs_a), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_as_e), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_cz_negate), (ag_ve_cz_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_out_mem_sel), (ag_ve_out_mem_sel), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_accum_sel), (ag_ve_accum_sel), (ag_ve_accum_sel), (ag_ve_accum_sel), (ag_ve_accum_sel), (ag_ve_accum_sel), (ag_ve_pre_accum_we),	
2 .ag_' 3 .ag_' 4 .ag_' 5 .ag_' 6 .ag_' 7 .ag_' 8 .ag_' 10 .ag_' 11 .ag_' 12 .ag_' 13); 14 .ag_' 15 .pa_cl_ 16 .upa_c 17 (18 .iag_ 19 .iag_ 20 .iag_ 21 .iag_ 22 .iag_ 23 .iag_ 24 .iag_ 24 .iag_	ve_bw_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_ex_negate ve_bce_flat_tst ve_out_mem_sel ve_out_addr ve_out_we ve_accum_sel ve_pre_accum_we I_ve cl_ve _ve_in_a0	(ag_ve_bw_negate), (ag_ve_cx_negate), (ag_ve_cy_negate), (ag_ve_cy_negate), (ag_ve_cw_negate), (ag_ve_bec_flat_tst), (ag_ve_bec_flat_tst), (ag_ve_out_mem_sel), (ag_ve_out_addr), (ag_ve_out_sel), (ag_ve_pre_accum_we) (ag_ve_pre_accum_we) (ag_ve_in_a0), (ag_ve_in_a1), (ag_ve_in_a2), (ag_ve_in_b0),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	iag ve a is wwww iag ve broadcast x iag ve abs a iag ve abs b iag ve abs c iag ve abs c iag ve as c iag ve as c iag ve as c iag ve as negate iag ve as negate iag ve as negate iag ve as negate iag ve bas negate iag ve bs negate iag ve bs negate iag ve bs negate iag ve bs negate iag ve c negate iag ve c c negate iag ve cy negate iag ve cy negate iag ve co negate iag ve cou negate iag ve out mem sel iag ve out mem sel iag ve out we iag ve accum sel	(ag_ve_a_is_www), (ag_ve_broadcast_x), (ag_ve_abs_a), (ag_ve_abs_b), (ag_ve_abs_b), (ag_ve_as_e), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_ar_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_br_negate), (ag_ve_cz_negate), (ag_ve_cz_negate), (ag_ve_cv_negate), (ag_ve_cv_negate), (ag_ve_out_mer_sel), (ag_ve_out_mer_sel), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_out_we), (ag_ve_out_we),	

```
.ove waddr
                                (ve_waddr),
                                                                                                                               .rbiu_vte_window_offset_sel (rbiu_vte_window_offset_sel),
 2
       .ove_veoc_vector_back_we
                                    (ve_veoc_vector_back_we),
                                                                                                                               .rbiu\_vte\_window\_offset\_en\_sel \qquad (rbiu\_vte\_window\_offset\_en\_sel),
       .ove_cliptemp_vector_we
                                  (ve_cliptemp_vector_we),
                                                                                                                               .rbiu\_vte\_persp\_corr\_dis\_sel \qquad (rbiu\_vte\_persp\_corr\_dis\_sel),
       .ove_inverse_we
                                 (ve_inverse_we),
                                                                                                                                                      (ag_ve_in_a0),
       .ove_wdata
                               (ve_wdata),
                                                                                                                               .ag_vte_jy
                                                                                                                                                      (ag_ve_in_al),
       .isclk
                            (sclk_pa)
                                                                                                                               .ag_vte_z
                                                                                                                                                      (ag_ve_in_a2),
                                                                                                                               .ag_vte_w
                                                                                                                                                       (ag_ve_in_a3),
                                                                                                                               .ag_vte_negate_ix
                                                                                                                                                        (ag_ve_ax_negate),
      pa_vte
                                                                                                                               .ag_vte_negate_jy
                                                                                                                                                         (ag_ve_ay_negate),
      upa_vte
                                                                                                                              .ag_vte_negate_z
                                                                                                                                                         (ag_ve_az_negate),
11
                                                                                                                              .ag_vte_negate_w
                                                                                                                                                          (ag_ve_aw_negate),
12
       .rbiu_vte_wdata
                                 (rbiu_wdata),
                                                                                                                              .ag_vte_vertex_store_indx (ag_vte_vertex_store_indx),
13
       .rbiu_vte_waddr
                                 (rbiu_waddr),
                                                                                                                               .ag_vte_opcode
                                                                                                                                                         (ag vte opcode),
14
       .rbiu_vte_raddr
                                 (rbiu_raddr),
                                                                                                                                                         (ag_vte_st_indx),
                                                                                                                               .ag vte st indx
15
                                (rbiu_we),
                                                                                                                        15
       .rbiu vte we
                                                                                                                                                     (vcm rcp rei d),
                                                                                                                               .rcp d
16
       .rbiu vte re
                               (rbiu re),
                                                                                                                        16
                                                                                                                                                     (vcm rcp rei a).
                                                                                                                               .гер а
17
                                (rbiu_vte_cpy),
                                                                                                                        17
       .rbiu vte cpy
                                                                                                                               .rcp xfc
                                                                                                                                                     (vem rep rei xfe),
18
       .rbiu vte xscale sel
                                  (rbiu vte xscale sel).
                                                                                                                        18
                                                                                                                               .vte rcp d
                                                                                                                                                      (vte rcp d).
19
       .rbiu vte xoffset sel
                                  (rbiu vte xoffset sel),
                                                                                                                        19
                                                                                                                               vte rbiu rdata
                                                                                                                                                       (vte rbiu rdata),
20
                                                                                                                        20
       rbiu vte vscale sel
                                  (rbiu vte vscale sel),
                                                                                                                               vte rcp rts
                                                                                                                                                       (vte rcp rts),
21
       .rbiu vte voffset sel
                                                                                                                        21
                                  (rbiu vte yoffset sel),
                                                                                                                               .vte d
                                                                                                                                                     (vte d),
22
       .rbiu vte zscale sel
                                  (rbiu vte zscale sel),
                                                                                                                        22
                                                                                                                               .vte opcode
                                                                                                                                                        (vte opcode),
23
       .rbiu vte zoffset sel
                                  (rbiu vte zoffset sel),
                                                                                                                        23
                                                                                                                               .vte_vertex_store_indx
                                                                                                                                                          (vte_vertex_store_indx),
24
       .rbiu_vte_cntl_sel
                                 (rbiu_vte_cntl_sel),
                                                                                                                        24
                                                                                                                               .vte_busy
                                                                                                                                                      (vte_busy),
       .rbiu_vte_vtx_cntl_sel
                                   (rbiu_vte_vtx_cntl_sel),
                                                                                                                        25
                                                                                                                                                    (srst),
                                            Page 45 of 63
                                                                                                                                                                    Page 46 of 63
                                                                                Ex. 2114 - pa.v
                                                                                                                                                                                                        Ex. 2114 - pa.v
                            (sclk_pa)
                                                                                                                                                    (sclk_pa)
 2
                                                                                                                         2
 4
      pa cl rei
                                                                                                                         4
                                                                                                                             pa cl repeng
      upa_cl_rei
                                                                                                                              upa_cl_rcpeng
 6
                                                                                                                         6
       .isc_wdata0
                               (ve wdata[31:0]),
                                                                                                                               ireset
                                                                                                                                                     (srst).
       .isc_rei_we
                               (ve_inverse_we),
                                                                                                                               .iport1 rts
                                                                                                                                                      (vte_rcp_rts),
10
       .ovmc_rei_rcp_d
                                 (vmc_rei_rcp_d),
                                                                                                                        10
                                                                                                                               .iport1 sign
                                                                                                                                                       (vte_rcp_d[31]),
11
       .ovmc_rei_rcp_rts
                                  (vmc_rei_rcp_rts),
                                                                                                                        11
                                                                                                                               .iport1_exp
                                                                                                                                                        (vte_rcp_d[30:23]),
       .ivcm_rcp_rei_rtr
12
                                 (vcm_rcp_rei_rtr),
                                                                                                                        12
                                                                                                                               .iport1_mant
                                                                                                                                                        (vte_rcp_d[22:0]),
13
                                                                                                                        13
14
       .ivcm_rcp_rei_d
                                 (vcm_rcp_rei_d),
                                                                                                                        14
                                                                                                                               .iport2_rts
                                                                                                                                                       (vmc_rei_rcp_rts),
15
       .ivcm_rcp_rei_a
                                 ({vcm_rcp_rei_a,1'b0}),
                                                                                                                               .iport2_sign
                                                                                                                                                        (vmc_rei_rcp_d[31]),
16
                                                                                                                               .iport2_exp
                                                                                                                                                        (vmc_rei_rcp_d[30:23]),
       .ivcm_rcp_rei_xfc
                                 (vcm_rcp_rei_xfc),
17
                                                                                                                        17
                                                                                                                               .iport2_mant
                                                                                                                                                        (vmc_rei_rcp_d[22:0]),
18
       .ovmb_cl_rei_r0vld
                                  (vmb_cl_rei_r0vld),
                                                                                                                        18
19
       .ovmb_cl_rei_rlvld
                                   (vmb_cl_rei_r1vld),
                                                                                                                        19
                                                                                                                               .oport2 rtr
                                                                                                                                                       (vcm rcp rei rtr),
20
       .ivbm cl rei r0r1clr
                                  (cl rei clear result),
                                                                                                                        20
21
                                                                                                                        21
                                                                                                                                                       (vcm rcp rei xfc),
                                                                                                                               .oout xfc
22
                                (rei_sc_r0data),
                                                                                                                        22
       .orei sc r0data
                                                                                                                                                         (vcm rcp rei d[31]),
                                                                                                                               .oout_recip_sign
23
       .orei_sc_r1data
                                (rei_sc_rldata),
                                                                                                                        23
                                                                                                                                                         (vcm rcp rei d[30:23]),
                                                                                                                               .oout recip exp
24
                                                                                                                        24
                                                                                                                               .oout recip mant
                                                                                                                                                          (vcm rcp rei d[22:0]),
25
       isrst
                            (srst),
                                                                                                                        25
                                                                                                                               .oout sel
                                                                                                                                                      (vcm rcp rei a),
                                            Page 47 of 63
                                                                                                                                                                    Page 48 of 63
                                                                               Ex. 2114 - pa.v
                                                                                                                                                                                                        Ex. 2114 - pa.v
```

				1			
1				1	.isxif_st_r_addr	(rbiu_raddr),	
2	.orcp_busy (),			2	.isxif_st_re	(rbiu_re),	
3				3	.isxif_st_sel	(rbiu_ccg_expcntmd_sel),	
4		6'h0),		4	.isxif_st_cpy	(rbiu_ccg_cpy),	
5	.odbug_data_out (0,		5			
6 7	inally (author			6 7	// vgt_to_ccgen fifo	(VCT DA alia a good a)	
8	.isclk (sclk_);	pa)		8	.ivgt_to_ccgen_fifo_write .ivgt_to_ccgen_fifo_active_ve	(VGT_PA_clip_v_send_q), erts (VGT_PA_clip_v_vec_size_q),	
9),			9	.ivgt_to_ccgen_fifo_state_var_		
10				10	88		
11	// shader export interface and cl	lip code generator		11	// sx0 receive fifo		
12	pa_sxifccg			12	.isx0_receive_fifo_write	(SX0_PA_input_data_write),	
13	upa_sxifccg			13	.isx0_receive_fifo_wrdata	(SX0_PA_input_data_wrdata),	
14	(14			
15	///////////////////////////////////////	111111111111111111111111111111111111111		15	// sx1 receive fifo		
16	// inputs			16	.isx1_receive_fifo_write	(SX1_PA_input_data_write),	
17	//////////////////////////////////////	(11111111111111111111111111111111111111		17	.isx1_receive_fifo_wrdata	(SX1_PA_input_data_wrdata),	
18	// clock and reset	(N)		18			
19		(sclk_pa),		19	// ccg state	(al ann atatat)	
20	.reset	(srst),		20 21	.iccg_state0	(cl_ccg_state0), (cl_ccg_state1),	
22	// ati state storage (sxif)			21 22	.iccg_state1	(cl_ccg_state1), (cl_ccg_state2),	
23	.isxif_st_w_data	(rbiu_wdata[26:20]),		23	.iccg_state3	(cl_ccg_state3),	
24	.isxif_st_w_addr	(rbiu_waddr),		24	.iccg_state4	(cl_ccg_state4),	
25	.isxif_st_we	(rbiu_we),		25	.iccg_state5	(cl_ccg_state5),	
		Page 49 of 63	Ex. 2114 - pa.v			Page 50 of 63	Ex. 2114 - pa.v
1	igag statak	(al agg etatas)		,	acrif stata7	(cvif ctata7)	
1 2	.iccg_state6	(cl_ccg_state6), (cl_ccg_state7).		1 2	.osxif_state7	(sxif_state7),	
2	.iccg_state6	(cl_ccg_state6), (cl_ccg_state7),		2		(sxif_state7),	
					.osxif_state7 // vgt_to_ccgen fifo .ovgt_to_ccgen fifo notfull		
3	.iccg_state7			2 3	// vgt_to_ccgen fifo	(sxif_state7), (PA_VGT_clip_v_d),	
2 3 4	.iccg_state7 // ccgen_to_clipcc/clip	(cl_ccg_state7),		2 3 4	// vgt_to_ccgen fifo		
2 3 4 5	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices),		2 3 4 5	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull		
2 3 4 5 6	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices),		2 3 4 5 6	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request	(PA_VGT_clip_v_d),	
2 3 4 5 6 7 8	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices),		2 3 4 5 6 7 8 9	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d),	
2 3 4 5 6 7 8 9	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipce_fifo_full // arbiter	(cl_ccg_outsm_clr_orig_vertices), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipcc_fifo_full),		2 3 4 5 6 7 8 9	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d),	
2 3 4 5 6 7 8 9 10	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipce_fifo_full // arbiter .iarb_to_ccgen_xfc	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipec_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d),	
2 3 4 5 6 7 8 9 10 11	.iccg_state7 // ccgen_to_clipce/clip .ioutsm_clr_orig_vertices .iccgen_to_clipce_fifo_full // arbiter .iarb_to_ccgen_xfc	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipec_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d),	
2 3 4 5 6 7 8 9 10 11 12 13	.iccg_state7 // ccgen_to_clipce/clip .ioutsm_clr_orig_vertices .iccgen_to_clipce_fifo_full // arbiter .iarb_to_ccgen_xfc	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipec_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11 12 13	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d),	
2 3 4 5 6 7 8 9 10 11 12 13	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipec_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11 12 13	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d),	
2 3 4 5 6 7 8 9 10 11 12 13	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipec_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11 12 13	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipcc_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11 12 13 14	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipcc_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_sp_id .opa_to_sx1_offset	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // state .osxif_st_r_data	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipcc_fifo_full), (arb_ccg_xfc),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	// vgt_to_ccgen fifo ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_offset .opa_to_sx1_offset .opa_to_sx1_offset .opa_to_sx1_aux	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // state .osxif_st_r_data	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	// vgt_to_ccgen fifo ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_offset .opa_to_sx1_offset .opa_to_sx1_offset .opa_to_sx1_aux	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.iccg_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // outputs // state .osxif_st_r_data // state to clipper .osxif_state0	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	// vgt_to_ccgen fifo ovgt_to_ccgen_fifo_notfull // sx0, request opa_to_sx0_req opa_to_sx0_sp_id opa_to_sx0_offset opa_to_sx0_aux opa_to_sx0_last // sx1, request opa_to_sx1_req opa_to_sx1_sp_id opa_to_sx1_offset opa_to_sx1_offset opa_to_sx1_aux opa_to_sx1_aux	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	// ccge_state7 // ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // outputs // state .osxif_st_r_data // state to clipper .osxif_state0 .osxif_state1 .osxif_state2 .osxif_state3	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_ccgen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20), (sxif_state0), (sxif_state1),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_offset .opa_to_sx1_aux .opa_to_sx1_aux .opa_to_sx1_last	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d), (u1_PA_SX_aux_d), (u1_PA_SX_last_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // outputs // state to clipper .osxif_state0 .osxif_state1 .osxif_state2 .osxif_state3 .osxif_state4	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_cegen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20), (sxif_state0), (sxif_state1), (sxif_state2), (sxif_state2), (sxif_state3), (sxif_state4),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_offset .opa_to_sx1_aux .opa_to_sx1_last // position memory .oposition_write	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d), (u1_PA_SX_aux_d), (u1_PA_SX_last_d),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // outputs // state to clipper .osxif_stare1 .osxif_state2 .osxif_state3 .osxif_state4 .osxif_state4	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_cegen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20), (sxif_state0), (sxif_state1), (sxif_state2), (sxif_state2), (sxif_state3), (sxif_state4), (sxif_state4),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_sp_id .opa_to_sx1_aux .opa_to_sx1_last // position memory .oposition_write .oposition_wrdata	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d), (u1_PA_SX_aux_d), (u1_PA_SX_last_d), (ccg_ag_pos_mem_we), (ccg_ag_pos_mem_wraddr),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // outputs // state to clipper .osxif_state0 .osxif_state1 .osxif_state2 .osxif_state3 .osxif_state4	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_cegen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20), (sxif_state0), (sxif_state1), (sxif_state2), (sxif_state2), (sxif_state3), (sxif_state4),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_offset .opa_to_sx1_aux .opa_to_sx1_last // position memory .oposition_write .oposition_wraddr	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d), (u1_PA_SX_aux_d), (u1_PA_SX_last_d), (ccg_ag_pos_mem_we), (ccg_ag_pos_mem_wraddr),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// ccgen_to_clipcc/clip .ioutsm_clr_orig_vertices .iccgen_to_clipcc_fifo_full // arbiter .iarb_to_ccgen_xfc // outputs // outputs // state to clipper .osxif_stare1 .osxif_state2 .osxif_state3 .osxif_state4 .osxif_state4	(cl_ccg_state7), (cl_ccg_outsm_clr_orig_vertices), (cl_ccg_cegen_to_clipce_fifo_full), (arb_ccg_xfc), (ccg_rbiu_rdata_26_downto_20), (sxif_state0), (sxif_state1), (sxif_state2), (sxif_state2), (sxif_state3), (sxif_state4), (sxif_state4),	Ex. 2114 - pa.v	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// vgt_to_ccgen fifo .ovgt_to_ccgen_fifo_notfull // sx0, request .opa_to_sx0_req .opa_to_sx0_sp_id .opa_to_sx0_offset .opa_to_sx0_aux .opa_to_sx0_last // sx1, request .opa_to_sx1_req .opa_to_sx1_sp_id .opa_to_sx1_sp_id .opa_to_sx1_aux .opa_to_sx1_last // position memory .oposition_write .oposition_wrdata	(PA_VGT_clip_v_d), (u0_PA_SX_req_d), (u0_PA_SX_sp_id_d), (u0_PA_SX_offset_d), (u0_PA_SX_aux_d), (u0_PA_SX_last_d), (u1_PA_SX_req_d), (u1_PA_SX_sp_id_d), (u1_PA_SX_offset_d), (u1_PA_SX_aux_d), (u1_PA_SX_aux_d), (u1_PA_SX_last_d), (ccg_ag_pos_mem_we), (ccg_ag_pos_mem_wraddr),	Ex. 2114 - pa.v

l .opoin	int_write (ccg_ag_pntsz_mem_we),		1	.reset (srst),	
		(ccg_ag_pntsz_mem_wraddr),		2	eset (,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	
-		(ccg_ag_pntsz_mem_wrdata),		3	// ati state storage		
4	_			4	.ist_w_data	(rbiu_wdata),	
5 // ccge	gen_to_clipcc/clip			5	.ist_w_addr	(rbiu_waddr),	
6 .occge	gen_to_clipcc_data	(ccg_cl_wrdata),		6	.ist_we	(rbiu_we),	
7 .occge	gen_to_clipcc_write	(ccg_cl_write),		7	.ist_r_addr	(rbiu_raddr),	
8				8	.ist_re ((rbiu_re),	
9 // arbi	piter			9	.ist_cpy	(rbiu_cl_cpy),	
10 .occge	gen_to_arb_data	(ccg_arb_data)		10	.ist_sel ((rbiu_cl_dx_clip_sp_def_sel),	
11);				11			
12				12	// state from sxif		
	p connections			13	.isxif_state0	(sxif_state0),	
	ag_cl_pntsz_mem_blocked =			14	.isxif_state1	(sxif_state1),	
	ag_cl_pntsz_mem_rdata = 'h();		15	.isxif_state2	(sxif_state2),	
16	nar			16	.isxif_state3	(sxif_state3),	
17 // clippe 18 pa_clip				17 18	.isxif_state4 .isxif_state5	(sxif_state4), (sxif_state5),	
18 pa_clip 19 upa_cli				19	.isxif_state6	(sxif_state5), (sxif_state6),	
20 (20	.isxif_state7	(sxif_state7),	
		(((((((((((((((((((((((((((((((((((((((21	= "	. = "	
22 // inpu				22	// ccg		
23 ///////	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	1111111111111111		23	.iccgen_to_clipcc_fifo_write	(ccg_cl_write),	
24 // cloc	ock and reset			24	.iccgen_to_clipcc_fifo_wrdata	(ccg_cl_wrdata),	
25 .clk	(sclk	_pa),		25			
		Page 53 of 63				Page 54 of 63	
			ix. 2114 - pa.v				Ex. 2114 - pa.v
1 //yet	t to clins				ichin to ma fifo notfull	(su clin ttr)	LA. 2114 - pu. v
	t_to_clips to clips fifo write			1 2	.iclip_to_ga_fifo_notfull .iclip_ga_be_fifo_notfull	(su_clip_rtr), (su_clip_barve_rtr),	Е.с. 2114 - ра. ч
2 .ivgt_t	_to_clips_fifo_write	(VGT_PA_elip_s_send_q),		1 2 3	.iclip_to_ga_fifo_notfull .iclip_ga_bc_fifo_notfull	(su_clip_rtr), (su_clip_baryc_rtr),	Lx. 2117 - pa. 1
2 .ivgt_1 3 .ivgt_1	_to_clips_fifo_write _to_clips_fifo_event	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q),		2			LX. 2114 - pa. 4
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1	_to_clips_fifo_write	(VGT_PA_elip_s_send_q),		2	.iclip_ga_bc_fifo_notfull		La. 2114 - pa. v
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1	_to_clips_fifo_write _to_clips_fifo_event _to_clips_fifo_prim_type	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q),		2 3 4	.iclip_ga_bc_fifo_notfull // from vte	(su_clip_baryc_rtr),	LX. 2117 - pa. 1
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6	_to_clips_fifo_write _to_clips_fifo_event _to_clips_fifo_prim_type	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q),		2 3 4 5	.iclip_ga_bc_fifo_notfull // from vte .ivte_out_vertex_store_indx	(su_clip_baryc_rtr), (vte_vertex_store_indx),	LX. 2117 - pu. 1
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6 7 // vgt_1	_to_clips_fifo_write _to_clips_fifo_event _to_clips_fifo_prim_type _to_clips_fifo_state_var_indx	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q),		2 3 4 5	.iclip_ga_bc_fifo_notfull // from vte .ivte_out_vertex_store_indx .ivte_out_opcode	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode),	LX. 2117 - pa. 1
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6 7 // vgt_8 .ivgt_1	_to_clips_fifo_write _to_clips_fifo_event _to_clips_fifo_prim_type _to_clips_fifo_state_var_indx _t_to_clipp	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q),		2 3 4 5 6 7	.iclip_ga_bc_fifo_notfull // from vte .ivte_out_vertex_store_indx .ivte_out_opcode	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d),	LX. 2117 - pa. 1
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6 7 // vgt_1 8 .ivgt_1 9 .ivgt_1 10 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_y	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q),		2 3 4 5 6 7 8	.iclip_ga_bc_fifo_notfull // from vte .ivte_out_vertex_store_indx .ivte_out_opcode .ivte_out_vector_data // from ag-ve_delay .iag_ve_out_vertex_store_indx	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]),	LX. Z117 - pu.
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 9 .ivgt_1 10 .ivgt_1 11 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_cto_clipp_fifo_deallocate_slot	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q),		2 3 4 5 6 7 8 9 10	iclip_ga_bc_fifo_notfull // from vte .ivte_out_vertex_store_indx .ivte_out_opcode .ivte_out_vector_data // from ag-ve_delay .iag_ve_out_vertex_store_indx .iag_ve_out_valid_bit_set	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set),	LX. Z117 - pa. 1
2 ivgt_ 3 ivgt_ 4 ivgt_ 5 ivgt_ 6 7 // vgt_ 8 ivgt_ 9 ivgt_ 10 ivgt_ 11 ivgt_ 12 ivgt_ 12 ivgt_	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_i to_clipp_fifo_deallocate_slot to_clipp_fifo_end_of_packet	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_eop_q),		2 3 4 5 6 7 8 9 10	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_velid_bit_set iag_ve_out_user_clip_indx	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx),	LX. Z117 - pa. 1
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 9 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 13 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_; to_clipp_fifo_deallocate_slot to_clipp_fifo_end_of_packet to_clipp_fifo_edge_flag	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edge_flags_q),		2 3 4 5 6 7 8 9 10 11 12 13	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_velid_bit_set iag_ve_out_valid_bit_set iag_ve_out_vertex_clip_indx iag_ve_out_vv_cc_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test),	LX. 2117 - pa. 1
2 .ivgt_1 3 .ivgt_1 4 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 14 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_; to_clipp_fifo_deallocate_slot to_clipp_fifo_end_of_packet to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edge_flags_q), (VGT_PA_clip_p_edge_flags_q), adx_0 (VGT_PA_clip_p_indx0_q),		2 3 4 5 6 7 8 9 10 11 12 13	iclip_ga_bc_fifo_notfull // from vte .ivte_out_vertex_store_indx .ivte_out_opcode .ivte_out_vector_data // from ag-ve_delay .iag_ve_out_vertex_store_indx .iag_ve_out_valid_bit_set .iag_ve_out_user_clip_indx .iag_ve_out_vv_cc_test .iag_ve_out_ucp_cc_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test), (ag_cl_ucp_cc_test),	LX. 2117 - pa. 1
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 14 .ivgt_1 15 .ivgt_1 15 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_si to_clipp_fifo_deallocate_slot to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edge_flags_q), (VGT_PA_clip_p_edge_flags_q), adx_0 (VGT_PA_clip_p_indx0_q), adx_1 (VGT_PA_clip_p_indx1_q),		2 3 4 5 6 7 8 9 10 11 12 13 14	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_valid_bit_set iag_ve_out_valid_bit_set iag_ve_out_vv_cc_test iag_ve_out_ucp_cc_test iag_ve_out_ucp_cc_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test), (ag_cl_ucp_cc_test), (ag_cl_bcc_cc_test),	LX. 2117 - pa.
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 14 .ivgt_1 15 .ivgt_1 16 .ivgt_1 16 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_; to_clipp_fifo_deallocate_slot to_clipp_fifo_end_of_packet to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edge_flags_q), (VGT_PA_clip_p_edge_flags_q), adx_0 (VGT_PA_clip_p_indx0_q), adx_1 (VGT_PA_clip_p_indx1_q),		2 3 4 5 6 7 8 9 10 11 12 13 14 15	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_valid_bit_set iag_ve_out_valid_bit_set iag_ve_out_user_clip_indx iag_ve_out_vv_cc_test iag_ve_out_ucp_cc_test iag_ve_out_bcc_cc_test iag_ve_out_ps_ucp_cc_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vvc_test), (ag_cl_ucp_cc_test), (ag_cl_pbc_cc_test), (ag_cl_ps_ucp_cc_test),	LX. Z117 - pu.
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 14 .ivgt_1 15 .ivgt_1 16 .ivgt_1 17	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_s to_clipp_fifo_deallocate_slot to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edge_flags_q), (VGT_PA_clip_p_edge_flags_q), adx_0 (VGT_PA_clip_p_indx0_q), adx_1 (VGT_PA_clip_p_indx1_q),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_valid_bit_set iag_ve_out_valid_bit_set iag_ve_out_vv_cc_test iag_ve_out_ucp_cc_test iag_ve_out_ucp_cc_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test), (ag_cl_ucp_cc_test), (ag_cl_bcc_cc_test),	LX. Z117 - pu.
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 15 .ivgt_1 16 .ivgt_1 17 18 // recip	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_null_primitive to_clipp_fifo_first_prim_of_si to_clipp_fifo_deallocate_slot to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), slot (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edge_flags_q), (VGT_PA_clip_p_edge_flags_q), adx_0 (VGT_PA_clip_p_indx0_q), adx_1 (VGT_PA_clip_p_indx1_q),		2 3 4 5 6 7 8 9 10 11 12 13 14 15	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_valid_bit_set iag_ve_out_valid_bit_set iag_ve_out_user_clip_indx iag_ve_out_vv_cc_test iag_ve_out_ucp_cc_test iag_ve_out_bcc_cc_test iag_ve_out_ps_ucp_cc_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vvc_test), (ag_cl_ucp_cc_test), (ag_cl_pbc_cc_test), (ag_cl_ps_ucp_cc_test),	L.C. Z.1.7 - pa. 1
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 15 .ivgt_1 16 .ivgt_1 17 18 // recit 19 .ivgt_1 19 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_prim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_state_var_inds to_clipp_fifo_deallocate_slot to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_edpe_flags_q), (VGT_PA_clip_p_edge_flags_q), dx_0 (VGT_PA_clip_p_indx0_q), dx_1 (VGT_PA_clip_p_indx1_q), dx_2 (VGT_PA_clip_p_indx2_q),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_valid_bit_set iag_ve_out_valid_bit_set iag_ve_out_user_clip_indx iag_ve_out_vv_cc_test iag_ve_out_ucp_cc_test iag_ve_out_bcc_cc_test iag_ve_out_ps_ucp_cc_test iag_ve_out_ps_engh_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vvc_test), (ag_cl_ucp_cc_test), (ag_cl_pbc_cc_test), (ag_cl_ps_ucp_cc_test),	LA. ZIIV pa. v
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_2 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 15 .ivgt_1 16 .ivgt_1 17 18 // recit 19 .ivgt_1 19 .ivgt_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_event to_clips_fifo_state_var_indx t_to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_i to_clipp_fifo_deallocate_slot to_clipp_fifo_edge_flag to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_data_valid_0	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_indx0_q), adx_0 (VGT_PA_clip_p_indx1_q), adx_1 (VGT_PA_clip_p_indx2_q), adx_2 (VGT_PA_clip_p_indx2_q), (vmb_cl_rei_r0vld),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_velid_bit_set iag_ve_out_velid_bit_set iag_ve_out_vv_cc_test iag_ve_out_vv_cc_test iag_ve_out_pc_cc_test iag_ve_out_ps_ucp_cc_test iag_ve_out_ps_ucp_cc_test iag_ve_out_ps_engh_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test), (ag_cl_ucp_cc_test), (ag_cl_pc_test), (ag_cl_ps_ucp_cc_test), (ag_cl_ps_ucp_cc_test), (ag_cl_ps_ucp_cc_test),	LA. ZIIV pa. v
2 .ivgt_1 3 .ivgt_1 5 .ivgt_1 6 7 // vgt_8 8 .ivgt_1 10 .ivgt_1 11 .ivgt_1 12 .ivgt_1 13 .ivgt_1 14 .ivgt_1 15 .ivgt_1 17 18 // recip 19 .iivy_1 20 .iivy_1 20 .iivy_1	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_event to_clips_fifo_brim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_i to_clipp_fifo_deallocate_slot to_clipp_fifo_ede_flag to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clip_data_valid_0 to_clip_data_valid_1	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_indx0_q), adx_0 (VGT_PA_clip_p_indx1_q), adx_1 (VGT_PA_clip_p_indx2_q), adx_2 (VGT_PA_clip_p_indx2_q), (vmb_cl_rei_r0vld),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_velid_bit_set iag_ve_out_velid_bit_set iag_ve_out_vv_cc_test iag_ve_out_vv_cc_test iag_ve_out_pc_cc_test iag_ve_out_ps_ucp_cc_test iag_ve_out_ps_ucp_cc_test iag_ve_out_ps_engh_test	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test), (ag_cl_ucp_cc_test), (ag_cl_pc_test), (ag_cl_ps_ucp_cc_test), (ag_cl_ps_ucp_cc_test), (ag_cl_ps_ucp_cc_test),	LA. 21.77 pl. v
2 .ivgt_ 3 .ivgt_ 4 .ivgt_ 5 .ivgt_ 6 7	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_event to_clips_fifo_trim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_i to_clipp_fifo_deallocate_slot to_clipp_fifo_deallocate_slot to_clipp_fifo_evente_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clip_data_valid_0 to_clip_data_valid_1	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_indx0_q), adx_0 (VGT_PA_clip_p_indx1_q), adx_1 (VGT_PA_clip_p_indx2_q), adx_2 (VGT_PA_clip_p_indx2_q), (vmb_cl_rei_r0vld),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_velid_bit_set iag_ve_out_velid_bit_set iag_ve_out_vv_cc_test iag_ve_out_vv_cc_test iag_ve_out_ps_cc_test iag_ve_out_ps_ec_test iag_ve_out_ps_ec_test iag_ve_out_ps_engh_test // from vector engine ive_out_test_rtn_status	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_vv_cc_test), (ag_cl_ucp_cc_test), (ag_cl_pc_test), (ag_cl_ps_ucp_cc_test), (ag_cl_ps_ucp_cc_test), (ag_cl_ps_ucp_cc_test),	LA. 2114 - pa. v
2 .ivgt_ 3 .ivgt_ 4 .ivgt_ 5 .ivgt_ 6 7	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_event to_clips_fifo_trim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_i to_clipp_fifo_deallocate_slot to_clipp_fifo_deallocate_slot to_clipp_fifo_evente_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clip_data_valid_0 to_clip_data_valid_1	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_eod_q), (VGT_PA_clip_p_indx0_q), dx_1 (VGT_PA_clip_p_indx1_q), dx_2 (VGT_PA_clip_p_indx2_q), (vmb_cl_rei_r0vld),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_vertex_store_indx iag_ve_out_vertex_store_indx iag_ve_out_vertex_store_indx iag_ve_out_ver_ce_test iag_ve_out_vv_cc_test iag_ve_out_pv_cc_test iag_ve_out_ps_cc_test iag_ve_out_ps_engh_test // from vector engine ive_out_test_rtn_status	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_uver_ctest), (ag_cl_uvp_cc_test), (ag_cl_uvp_cc_test), (ag_cl_ps_uvp_cc_test), (ag_cl_ps_uvp_cc_test), (ag_cl_ps_engh_test), (vmb_ve_tst_rtn_stat),	
2	to_clips_fifo_write to_clips_fifo_event to_clips_fifo_event to_clips_fifo_trim_type to_clips_fifo_state_var_indx t_to_clipp to_clipp_fifo_write to_clipp_fifo_mull_primitive to_clipp_fifo_first_prim_of_i to_clipp_fifo_deallocate_slot to_clipp_fifo_deallocate_slot to_clipp_fifo_evente_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clipp_fifo_vertex_store_in to_clip_data_valid_0 to_clip_data_valid_1	(VGT_PA_clip_s_send_q), (VGT_PA_clip_s_event_q), (VGT_PA_clip_s_type_q), (VGT_PA_clip_s_state_q), (VGT_PA_clip_p_send_q), (VGT_PA_clip_p_null_prim_q), (VGT_PA_clip_p_new_vtx_vect_q), (VGT_PA_clip_p_dealloc_q), (VGT_PA_clip_p_eop_q), (VGT_PA_clip_p_eod_q), (VGT_PA_clip_p_indx0_q), dx_1 (VGT_PA_clip_p_indx1_q), dx_2 (VGT_PA_clip_p_indx2_q), (vmb_cl_rei_r0vld),		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	iclip_ga_bc_fifo_notfull // from vte ivte_out_vertex_store_indx ivte_out_opcode ivte_out_vector_data // from ag-ve_delay iag_ve_out_vertex_store_indx iag_ve_out_vertex_store_indx iag_ve_out_vertex_store_indx iag_ve_out_vertex_store_indx iag_ve_out_ver_ce_test iag_ve_out_vv_cc_test iag_ve_out_pv_cc_test iag_ve_out_ps_cc_test iag_ve_out_ps_engh_test // from vector engine ive_out_test_rtn_status	(su_clip_baryc_rtr), (vte_vertex_store_indx), (vte_opcode), (vte_d), (ag_cl_vertex_store_indx[3:0]), (ag_cl_valid_bit_set), (ag_cl_user_clip_indx), (ag_cl_uver_ctest), (ag_cl_uvp_cc_test), (ag_cl_bc_cc_test), (ag_cl_ps_up_cc_test), (ag_cl_ps_engh_test), (vmb_ve_tst_rtn_stat), (ag_to_clip_point_size),	

```
// outputs
                                                                                                                              // to clip_to_ga fifo
 2
       .oclip_to_ga_fifo_write
                                                                                                                                                                  (clip_su_rts),
       // state
                                                                                                                               .oclip_to_ga_point_size
                                                                                                                                                                  (clip_su_pt_size),
       .ost_r_data
                                     (cl_rbiu_rdata),
                                                                                                                               .oclip_to_ga_position_x_0
                                                                                                                                                                   (clip_su_x0),
                                                                                                                               .oclip_to_ga_position_x_l
                                                                                                                                                                    (clip_su_x1),
       // vgt_to_clips
                                                                                                                               .oclip_to_ga_position_x_2
                                                                                                                                                                    (clip_su_x2),
       .ovgt\_to\_clips\_fifo\_notfull
                                          (PA_VGT_clip_s_d),
                                                                                                                               .oclip_to_ga_position_y_0
                                                                                                                                                                    (clip_su_y0),
                                                                                                                               .oclip_to_ga_position_y_l
                                                                                                                                                                    (clip_su_y1),
       // vgt_to_clipp
                                                                                                                               .oclip_to_ga_position_y_2
                                                                                                                                                                    (clip_su_y2),
                                          (PA_VGT_clip_p_d),
       .ovgt\_to\_clipp\_fifo\_notfull
                                                                                                                               .oclip_to_ga_position_z_0
                                                                                                                                                                   (clip_su_z0),
11
                                                                                                                              .oclip_to_ga_position_z_l
                                                                                                                                                                   (clip_su_z1),
12
                                                                                                                              .oclip_to_ga_position_z_2
                                                                                                                                                                   (clip_su_z2),
13
       .occgen_to_clipcc_fifo_full
                                           (cl_ccg_ccgen_to_clipcc_fifo_full),
                                                                                                                                                                    (clip_su_w0),
                                                                                                                              oclip to ga position w 0
14
       .occg_state0
                                     (cl_ccg_state0),
                                                                                                                               .oclip_to_ga_position_w_l
                                                                                                                                                                    (clip_su_w1),
15
                                     (cl ccg statel),
                                                                                                                        15
                                                                                                                               oclip to ga position w 2
                                                                                                                                                                    (clip su w2),
       .occg state1
16
       .occg state2
                                     (cl ccg state2),
                                                                                                                         16
                                                                                                                               oclip to ga edge flag 0
                                                                                                                                                                   (clip su ef0),
17
                                                                                                                        17
                                                                                                                                                                   (clip_su_ef1),
       .occg state3
                                     (cl ccg state3),
                                                                                                                               oclip to ga edge flag 1
18
       .occg state4
                                     (cl_ccg_state4),
                                                                                                                        18
                                                                                                                               oclip to ga edge flag 2
                                                                                                                                                                   (clip su ef2),
19
       .occg state5
                                     (cl ccg state5),
                                                                                                                        19
                                                                                                                               .oclip to ga param cache indx 0
                                                                                                                                                                       (clip_su_attr_indx0),
20
       .occg state6
                                     (cl ccg state6),
                                                                                                                        20
                                                                                                                               .oclip to ga param cache indx 1
                                                                                                                                                                       (clip su attr indx1).
21
       .occg_state7
                                     (cl_ccg_state7),
                                                                                                                        21
                                                                                                                               .oclip\_to\_ga\_param\_cache\_indx\_2
                                                                                                                                                                       (clip_su_attr_indx2),
22
                                                                                                                        22
                                                                                                                               .oclip_to_ga_prim_type
                                                                                                                                                                   (clip su type),
23
       // to ccgen
                                                                                                                        23
                                                                                                                               .oclip_to_ga_state_var_indx
                                                                                                                                                                    (clip_su_st_indx),
24
       .ooutsm_clr_orig_vertices
                                           (cl_ccg_outsm_clr_orig_vertices),
                                                                                                                        24
                                                                                                                               .oclip_to_ga_deallocate_slot
                                                                                                                                                                    (clip_su_dealloc_slot),
25
                                                                                                                               .oclip_to_ga_null_primitive
                                                                                                                                                                    (clip_su_null_prim),
                                            Page 57 of 63
                                                                                                                                                                    Page 58 of 63
                                                                                Ex. 2114 - pa.v
                                                                                                                                                                                                        Ex. 2114 - pa.v
       .oclip_to_ga_clipped_prim
                                                                                                                                                                   (clip_to_ag_point_buf_re),
                                           (clip su clipped),
                                                                                                                               .oclip to ag point buf re
 2
                                                                                                                                                                     (clip to ag point buf raddr),
       oclip to ga first prim of slot
                                            (clip su fpov),
                                                                                                                               .oclip to ag point buf raddr
       .oclip\_to\_ga\_end\_of\_packet
                                            (clip_su_eop),
                                                                                                                               // busy
       .oclip_to_ga_event
                                        (clip su event),
       .oclip_to_ga_event_id
                                         (clip_su_event_id)
                                                                                                                               .oclipper busy
                                                                                                                                                               (clipper busy)
                                                                                                                         6
       .oclip\_ga\_bc\_fifo\_write
                                          (clip_su_baryc_rts),
       .oclip\_ga\_bc\_baryc\_coord\_x\_0
                                              (clip su i0),
                                                                                                                             // temp connections until clipper has these
       .oclip_ga_bc_baryc_coord_x_l
                                              (clip_su_il),
                                                                                                                             //assign clip su k0 = 32'h00000000;
10
       .oclip_ga_bc_baryc_coord_x_2
                                              (clip_su_i2),
                                                                                                                             //assign clip_su_k1 = 32'h00000000;
11
       .oclip\_ga\_bc\_baryc\_coord\_y\_0
                                              (clip_su_j0),
                                                                                                                        11 //assign clip_su_k2 = 32'h00000000;
12
       .oclip_ga_bc_baryc_coord_y_l
                                              (clip_su_j1),
                                                                                                                             //assign clip_su_baryc_rts = 1'b0;
13
                                              (clip_su_j2),
                                                                                                                        13
       .oclip_ga_bc_baryc_coord_y_2
14
       .oclip_ga_bc_baryc_coord_z_0
                                              (clip_su_k0),
                                                                                                                         14 // setup unit
15
                                             (clip_su_k1),
       .oclip_ga_bc_baryc_coord_z_l
                                                                                                                             pa_su upa_su (
16
       .oclip\_ga\_bc\_baryc\_coord\_z\_2
                                              (clip_su_k2),
                                                                                                                               // outputs
17
18
                                                                                                                         18
                                                                                                                                     .PA_SC_p0_d(PA_SC_p0_d),
19
       .oclip_to_arb_data_ve_valid
                                            (cl_arb_ve_valid),
                                                                                                                                     .PA_SC_pl_d(PA_SC_pl_d),
20
                                                                                                                        20
                                                                                                                                     .PA SC p2 d(PA SC p2 d),
       .oclip to arb data
                                        (cl arb data),
21
                                                                                                                        21
                                                                                                                                     .PA SC p3 d(PA SC p3 d),
22
                                                                                                                        22
       // reciprocal engine
                                                                                                                                     .PA SC p4 d(PA SC p4 d),
23
                                                                                                                        23
       oclip to inv data reset valids
                                             (cl rei clear result).
                                                                                                                                     .PA SC xy0 d(PA SC xy0 d)
24
                                                                                                                        24
                                                                                                                                     .PA SC xy1 d(PA SC xy1 d).
25
       // ag point size memory
                                                                                                                                     .PA SC xy2 d(PA SC xy2 d).
                                            Page 59 of 63
                                                                                                                                                                    Page 60 of 63
                                                                                Ex. 2114 - pa.v
                                                                                                                                                                                                        Ex. 2114 - pa.v
```

```
.PA\_SC\_zminmax\_d(PA\_SC\_zminmax\_d),
                                                                                                                                           . SC\_PA\_early frz\_q (SC\_PA\_early frz\_q),
 2
             .PA\_SC\_cntl\_d(PA\_SC\_cntl\_d),
                                                                                                                               2
                                                                                                                                           .su\_busy(su\_busy),
             .PA\_SC\_valid\_d(PA\_SC\_valid\_d),
                                                                                                                                           .clip\_su\_pt\_size(clip\_su\_pt\_size),
             .PA_SC_phase_d(PA_SC_phase_d),
                                                                                                                                           .clip\_su\_x0(clip\_su\_x0),
            .PA_SC_v0_indx_d(PA_SC_v0_indx_d),
                                                                                                                                           .clip_su_x1(clip_su_x1),
             .su_clip_rtr(su_clip_rtr),
                                                                                                                                           .clip_su_x2(clip_su_x2),
             .su_clip_baryc_rtr(su_clip_baryc_rtr),
                                                                                                                                           .clip_su_y0(clip_su_y0),
             .su_rbiu_rdata(su_rbiu_rdata),
                                                                                                                                           .clip_su_y1(clip_su_y1),
              // inputs
                                                                                                                                           .clip_su_y2(clip_su_y2),
10
                                                                                                                                           .clip_su_z0(clip_su_z0),
11
             .sclk(sclk_pa),
                                                                                                                              11
                                                                                                                                           .clip_su_z1(clip_su_z1),
12
             .rbiu_su_wdata(rbiu_wdata),
                                                                                                                              12
                                                                                                                                           .clip_su_z2(clip_su_z2),
13
             .rbiu_su_we(rbiu_we),
                                                                                                                              13
                                                                                                                                           .clip_su_w0(clip_su_w0),
14
             .rbiu_su_re(rbiu_re),
                                                                                                                                           .clip_su_w1(clip_su_w1),
15
             .rbiu su waddr(rbiu waddr).
                                                                                                                              15
                                                                                                                                           .clip su w2(clip su w2),
16
             .rbiu su raddr(rbiu raddr),
                                                                                                                              16
                                                                                                                                           .clip su ef0(clip su ef0),
17
                                                                                                                              17
             .rbiu_su_cpy(rbiu_su_cpy),
                                                                                                                                           .clip su efl(clip su efl),
18
             .rbiu su expand lw sel(rbiu su expand lw sel),
                                                                                                                              18
                                                                                                                                           .clip su ef2(clip su ef2),
19
             .rbiu_su_imp_exp_sel(rbiu_su_imp_exp_sel),
                                                                                                                              19
                                                                                                                                           .clip_su_i0(clip_su_i0),
20
             .rbiu_su_draw_init_sel(rbiu_su_draw_init_sel),
                                                                                                                              20
                                                                                                                                           .clip_su_il(clip_su_il),
21
             .rbiu su status sel(rbiu su status sel),
                                                                                                                             21
                                                                                                                                           .clip_su_i2(clip_su_i2),
22
             .rbiu_su_point_size_sel(rbiu_su_point_size_sel),
                                                                                                                             22
                                                                                                                                           .clip_su_j0(clip_su_j0),
23
             .rbiu_su_point_min_max_sel(rbiu_su_point_min_max_sel),
                                                                                                                             23
                                                                                                                                           .clip_su_jl(clip_su_jl),
24
             .rbiu\_su\_line\_cntl\_sel(rbiu\_su\_line\_cntl\_sel),
                                                                                                                             24
                                                                                                                                           .clip\_su\_j2(clip\_su\_j2),
25
             .rbiu\_su\_sc\_mode\_cntl\_sel(rbiu\_su\_sc\_mode\_cntl\_sel),\\
                                                                                                                              25
                                                                                                                                           .clip_su_k0(clip_su_k0),
                                              Page 61 of 63
                                                                                                                                                                            Page 62 of 63
                                                                                    Ex. 2114 - pa.v
                                                                                                                                                                                                                  Ex. 2114 - pa.v
```

```
.clip_su_k1(clip_su_k1),
 2
              .clip_su_k2(clip_su_k2),
              .clip\_su\_attr\_indx0(clip\_su\_attr\_indx0),
              .clip\_su\_attr\_indx1(clip\_su\_attr\_indx1),
              .clip\_su\_attr\_indx2(clip\_su\_attr\_indx2),
              .clip\_su\_type(clip\_su\_type),
              .clip\_su\_st\_indx(clip\_su\_st\_indx),
              .clip\_su\_dealloc\_slot(clip\_su\_dealloc\_slot),
              .clip\_su\_null\_prim(clip\_su\_null\_prim),
10
              .clip\_su\_clipped(clip\_su\_clipped),
11
              .clip\_su\_fpov(clip\_su\_fpov),
12
              .clip_su_eop(clip_su_eop),
13
              .clip_su_event(clip_su_event),
14
              .clip_su_event_id(clip_su_event_id),
15
              .clip_su_rts(clip_su_rts),
16
              .clip\_su\_baryc\_rts(clip\_su\_baryc\_rts)
17 );
18
19 endmodule // pa
                                                 Page 63 of 63
                                                                                         Ex. 2114 - pa.v
```

```
1 'include "header.v"
                                                                                                        1 rbiu_ag_ucp0_sel,
                                                                                                        2 rbiu ag ucp1 sel,
                                                                                                         3 rbiu_ag_ucp2_sel,
4 \hspace{0.5cm} /\hspace{0.1cm}/\hspace{0.1cm} \$ Id: /\hspace{-0.1cm}/\hspace{0.1cm} depot/r400/devel/parts\_lib/src/gfx/pa/pa\_ag.v\#19~\$
5 //
                                                                                                         5 rbiu ag ucp4 sel.
                                                                                                         6 rbiu_ag_ucp5_sel,
7 //
                                                                                                        7 rbiu_ag_gb_sel,
                                                                                                        8 rbiu_ag_pntsz_sel,
9 // Notes: This file is the pa_ag address generator
                                                                                                        10 rbiu we.
11 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                        11 rbiu wa,
12 // © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
                                                                                                        12 rbiu_wd,
13 //
                                                                                                        13 rbiu_cpy,
14 // All rights reserved. This notice is intended as a precaution against
                                                                                                        14 rbiu_re,
15 // inadvertent publication and does not imply publication or any waiver
                                                                                                        15
                                                                                                            ag_rbiu_rd,
17 // year of creation of the work.
                                                                                                        17 ccg_to_arb_data,
18 //
                                                                                                        18 arb_ccg_xfc,
20
                                                                                                        20 clip_ve_valid,
21
                                                                                                        21 clip to arb data,
22 module pa_ag (
                                                                                                        22 arb_clip_xfc,
23 sclk, srst,
                                                                                                        23
                                                                                                        24 clip_to_ag_point_buf_re,
25 rbiu_ag_dx_clip_sp_def_sel,
                                                                                                        25 clip_to_ag_point_buf_raddr,
                                      Page 1 of 123
                                                                                                                                              Page 2 of 123
                                                                                                                                                                          Ex. 2115 - pa ag.v
                                                                 Ex. 2115 - pa ag.v
1 ag_to_clip_point_size,
                                                                                                        1 // ag_vte_ix, ag_vte_jy,
                                                                                                        2 // ag_vte_z, ag_vte_w,
3 pos_pntsz_ag_mem_data,
                                                                                                        3 // ag_vte_negate_ix, ag_vte_negate_jy,
5 pntsz_mem_waddr,
                                                                                                        5 ag vte opcode,
                                                                                                         6 ag_vte_st_indx,
 6 pos_mem_we,
                                                                                                        7 ag_vte_vertex_store_indx,
9 inv_ret_sc_data,
                                                                                                        9 ag ve opcode,
                                                                                                        10 ag_ve_in_a0, ag_ve_in_a1, ag_ve_in_a2, ag_ve_in_a3,
11 ve_cliptemp_vector_we,
                                                                                                        11 \qquad ag\_ve\_in\_b0, \qquad ag\_ve\_in\_b1, \qquad ag\_ve\_in\_b2, \quad ag\_ve\_in\_b3,
12 ve_veoc_vector_back_we,
                                                                                                        12 ag_ve_a_is_wwww, ag_ve_broadcast_x,
                                                                                                        13 ag_ve_abs_a, ag_ve_abs_b, ag_ve_abs_c,
14 ve_wdata,
                                                                                                        14 ag_ve_ax_negate, ag_ve_ay_negate, ag_ve_az_negate, ag_ve_aw_negate,
15
                                                                                                        15 ag_ve_bx_negate, ag_ve_by_negate, ag_ve_bz_negate, ag_ve_bw_negate,
16 agve_dly_valid_op,
                                                                                                        16 ag_ve_cx_negate, ag_ve_cy_negate, ag_ve_cz_negate, ag_ve_cw_negate,
17 agve_dly_vertex_store_indx,
                                                                                                        17 ag_ve_bcc_flat_tst, ag_ve_out_mem_sel, ag_ve_out_addr,
18 agve dly valid bit set,
                                                                                                        18 ag_ve_out_we, ag_ve_accum_sel, ag_ve_pre_accum_we
19 agve_dly_user_clip_indx,
20 agve_dly_vv_cc_test,
                                                                                                        20
                                                                                                        21 `include "pa_ag_pkg.v"
21 agve dly ucp cc test,
22 agve_dly_bcc_cc_test,
                                                                                                        23 // ****** PARAMETERS **************
                                                                                                        24 parameter u0_clptmp_ADDR_WIDTH = 6;
24 agve_dly_ps_engh_test,
                                                                                                        25 parameter u0_clptmp_DATA_WIDTH = 32;
                                      Page 3 of 123
                                                                                                                                             Page 4 of 123
                                                                 Ex. 2115 - pa ag.v
                                                                                                                                                                          Ex. 2115 - pa ag.v
```

ATI 2115 LG v. ATI IPR2015-00326

```
u0 clptmp WORDS = 35;
                                                                                                 1 parameter u0_stve_DEBUG = 0;
 1 parameter
 2 parameter
               u0 clptmp DEBUG = 0;
                                                                                                 2 parameter
                                                                                                                ul stve ADDR WIDTH = 6;
               u1\_clptmp\_ADDR\_WIDTH = 6;
                                                                                                    parameter
     parameter
                                                                                                                ul stve DATA WIDTH = 32;
               u1_clptmp_DATA_WIDTH = 32;
                                                                                                                ul stve WORDS = 64;
               ul_clptmp_WORDS = 35;
                                                                                                                ul stve DEBUG = 0;
               ul_clptmp_DEBUG = 0;
                                                                                                                u2_stve_ADDR_WIDTH = 6;
                u2\_clptmp\_ADDR\_WIDTH = 6;
                                                                                                                u2_stve_DATA_WIDTH = 32;
               u2_clptmp_DATA_WIDTH = 32;
                                                                                                              u2_stve_WORDS = 64;
               u2_clptmp_WORDS = 35;
                                                                                                              u2_stve_DEBUG = 0;
               u2\_clptmp\_DEBUG \quad = 0;
                                                                                                              u3_stve_ADDR_WIDTH = 6;
11 parameter
               u3_clptmp_ADDR_WIDTH = 6;
                                                                                                11 parameter u3_stve_DATA_WIDTH = 32;
12 parameter
               u3_clptmp_DATA_WIDTH = 32;
                                                                                                12 parameter u3_stve_WORDS = 64;
13 parameter
               u3_clptmp_WORDS = 35;
                                                                                                13 parameter u3_stve_DEBUG = 0;
14 parameter
               u3_clptmp_DEBUG = 0;
                                                                                                15 parameter
               u pntsz ADDR WIDTH = 6;
               u pntsz DATA WIDTH = 32;
                                                                                                16 input sclk;
16 parameter
17 parameter
               u pntsz WORDS = 64:
                                                                                                17 input
18 parameter
               u pntsz DEBUG = 0:
                                                                                                18
                                                                                                19 parameter
               u pos ADDR WIDTH = 6;
                                                                                                20 input rbiu_ag_dx_clip_sp_def_sel;
20 parameter
               u pos DATA WIDTH = 128;
21 parameter
               u pos WORDS = 64;
                                                                                                21 input [3:0] rbiu_ag_ucp0_sel;
22 parameter
               u pos DEBUG = 0;
                                                                                                22 input [3:0] rbiu_ag_ucpl_sel;
23 parameter
               u0 stve ADDR WIDTH = 6;
                                                                                                23 input [3:0] rbiu_ag_ucp2_sel;
24 parameter
               u0_stve_DATA_WIDTH = 32;
                                                                                                24 input [3:0] rbiu_ag_ucp3_sel;
               u0_stve_WORDS = 64;
                                                                                                25 input [3:0] rbiu_ag_ucp4_sel;
                                    Page 5 of 123
                                                                                                                                    Page 6 of 123
                                                             Ex. 2115 - pa_ag.v
                                                                                                                                                             Ex. 2115 - pa_ag.v
 1 input [3:0] rbiu_ag_ucp5_sel;
                                                                                                 2 // ******* position and pntsz mem **************
 3 input [3:0] rbiu_ag_gb_sel;
                                                                                                    input [127:0] pos_pntsz_ag_mem_data;
 4 input [3:0] rbiu_ag_pntsz_sel;
                                                                                                    input [5:0] pos_mem_waddr;
                                                                                                    input [5:0] pntsz_mem_waddr;
 6 input
            rbiu we;
                                                                                                          pos_mem_we;
     input [2:0] rbiu wa:
                                                                                                             pntsz_mem_we;
 8 input [31:0] rbiu wd
     input rbiu_cpy;
                                                                                                 9 // ****** interface with reciprocal engine *******
                                                                                                10 input [63:0] inv_ret_sc_data;
            rbiu_re;
11
     output [31:0] ag_rbiu_rd;
                                                                                                11
12
                                                                                                12 // ****** interface with ve output ***********
                                                                                                13 input [3:0] ve_cliptemp_vector_we;
14 // ******** interface with the ccg ************
                                                                                                 14 input [3:0] ve_veoc_vector_back_we;
15 input [15:0] ccg_to_arb_data; //Data from the ccg
                                                                                                15 input [5:0] ve_waddr;
     output arb_ccg_xfc;
                                                                                                 16 input [127:0] ve_wdata;
17
    // ******* interface with the clipper ********
                                                                                                18
19 input
                                                                                                19 output [2:0] ag_ve_opcode;
           clip ve valid;
20 input [29:0] clip to arb data;
                                                                                                20 output [31:0] ag ve in a0;
21 output arb clip xfc;
                                                                                                21 output [31:0] ag ve in al;
                                                                                                22 output [31:0] ag_ve_in_a2;
23 input clip_to_ag_point_buf_re;
                                                                                                23 output [31:0] ag_ve_in_a3;
24 input [5:0] clip_to_ag_point_buf_raddr;
                                                                                                24 output [31:0] ag ve in b0;
25 output [31:0] ag_to_clip_point_size;
                                                                                                25 output [31:0] ag ve in b1;
                                                                                                                                    Page 8 of 123
                                   Page 7 of 123
                                                             Ex. 2115 - pa_ag.v
                                                                                                                                                             Ex. 2115 - pa_ag.v
```

```
1 output [31:0] ag_ve_in_b2;
 2 output [31:0] ag_ve_in_b3;
                                                                                                      3 output ag_ve_a_is_www;
                                                                                                      3 output agve dly valid op;
              ag_ve_broadcast_x;
                                                                                                      4 output [5:0] agve_dly_vertex_store_indx;
             ag_ve_abs_a;
                                                                                                       5 output [1:0] agve_dly_valid_bit_set;
              ag_ve_abs_b;
                                                                                                         output [3:0] agve_dly_user_clip_indx;
             ag_ve_abs_c;
                                                                                                       7 output agve_dly_vv_cc_test;
             ag_ve_ax_negate;
                                                                                                       8 output agve_dly_ucp_cc_test;
 9 output
                                                                                                       9 output agve_dly_bcc_cc_test;
             ag_ve_ay_negate;
10 output
             ag_ve_az_negate;
                                                                                                      10 output agve_dly_ps_ucp_cc_test;
11 output
                                                                                                      11 output agve_dly_ps_engh_test;
             ag_ve_aw_negate;
12 output
             ag_ve_bx_negate;
                                                                                                      13 output
             ag ve by negate;
14 output
                                                                                                      14 //output [31:0] ag_vte_ix; // i/x Vector Data
             ag ve bz negate;
                                                                                                      15 //output [31:0] ag_vte_jy; // j/y Vector Data
15 output
             ag ve bw negate;
                                                                                                      16 //output [31:0] ag_vte_w; // w Vector Data
16 output
             ag ve cx negate;
17 output
                                                                                                      17 //output [31:0] ag_vte_z; // z Vector Data
              ag ve cy negate;
18 output
                                                                                                      18 //output [0:0] ag_vte_negate_ix; // Negate i/x Vector Data
              ag ve cz negate;
19 output
             ag_ve_cw_negate;
                                                                                                      19 //output [0:0] ag_vte_negate_jy; // Negated j/y Vector Data
                                                                                                      20 //output [0:0] ag vte negate w; // Negate w Vector Data
20 output
             ag ve bcc flat tst;
                                                                                                      21 //output [0:0] ag_vte_negate_z; // Nagate z Vector Data
21 output [2:0] ag_ve_out_mem_sel;
22 output [5:0] ag_ve_out_addr;
                                                                                                      22 output [1:0] ag_vte_vertex_store_indx;
23 output [3:0] ag_ve_out_we;
                                                                                                      23 output [2:0] ag_vte_opcode; // Opcode
24 output ag_ve_accum_sel;
                                                                                                      24 output [2:0] ag_vte_st_indx; // Context ID
25 output [3:0] ag_ve_pre_accum_we;
                                     Page 9 of 123
                                                                                                                                          Page 10 of 123
                                                                Ex. 2115 - pa_ag.v
                                                                                                                                                                      Ex. 2115 - pa_ag.v
     1 // pos read port signal declarations
 2 // Declare and map internal ccg interface signal names
                                                                                                      2 reg pos re;
 3 reg ccg_ve_cc_valid;
                                                                                                                  pos re r1;
 4 reg ccg_ve_valid;
                                                                                                      4 reg [5:0] pos raddr;
 5 reg [1:0] ccg_sm_state_indx;
                                                                                                      5 wire [127:0] pos rdata;
                                                                                                       6 reg [127:0] pos rdata r2;
 6 reg [2:0] ccg_state_var_indx;
 7 reg [5:0] ccg_vertex_store_indx;
 8 reg [2:0] ccg_ve_ucp_indx;
                                                                                                      8 // pntsz read port signal declarations
                                                                                                      9 reg pntsz_re;
10 // Declare and map internal clipper interface signal names
                                                                                                      10 reg pntsz_re_r1;
11 reg [3:0] clip_plane_indx;
                                                                                                      11 reg [5:0] pntsz_raddr;
12 reg [5:0] clip_dst_vertex_indx;
                                                                                                      12 wire [31:0] pntsz_rdata;
                                                                                                      13 reg [31:0] pntsz_rdata_r2;
13 reg [6:0] clip_src_vertex_indx;
14 reg clip_src_vertex_type;
                                                                                                      15 // stve read port signal declarations
15 reg clip_ve_ucp_valid;
16 reg [6:0] clip_sm_state_indx;
                                                                                                      16 reg stve_re_r0;
17 reg [2:0] clip_state_var_indx;
                                                                                                      17 reg stve_re_r1;
18
                                                                                                      18 reg [5:0] stve_raddr;
19 // ve_veoc_vector_back read port signal declarations
                                                                                                      19 wire [127:0] stve_rdata;
                                                                                                      20 reg [127:0] stve_rdata_r2;
20 reg ve veoc vector back re;
21 reg
                                                                                                     21 reg nxt stve re;
           ve veoc vector back re r1;
22 reg [5:0] ve_veoc_vector_back_raddr;
                                                                                                      22 reg [5:3] nxt_state_type;
23 wire [127:0] ve_veoc_vector_back_rdata;
                                                                                                      23 reg [5:3] state_type_r0;
24 reg [127:0] ve_veoc_vector_back_rdata_r2;
25
                                                                                                      25 // stve write port signal declarations
                                     Page 11 of 123
                                                                                                                                          Page 12 of 123
                                                                Ex. 2115 - pa_ag.v
                                                                                                                                                                      Ex. 2115 - pa_ag.v
```

```
l reg [5:0] stve_wa;
                                                                                                                   1 wire [2:0] agrd ucp1 write ptr;
 2 reg [3:0] stve_we;
                                                                                                                  3 wire [2:0] ucp2 rd off;
 4 // ag signal declarations
                                                                                                                  4 wire nxt_ucp2_write_after_cpy;
 5 reg [2:0] nxt_arb_state_var_indx;
                                                                                                                   5 reg ucp2_write_after_cpy;
 6 \hspace{0.5cm} \text{reg} \hspace{0.5cm} [2\text{:}0] \hspace{0.1cm} \text{arb\_state\_var\_indx\_r0};
                                                                                                                   6 wire [2:0] nxt_ucp2_write_ptr;
      reg [2:0] arb_ucp_indx;
                                                                                                                   7 reg [2:0] ucp2_write_ptr;
      reg [127:0] ve_cliptemp_vec;
                                                                                                                   8 wire [2:0] agrd_ucp2_write_ptr;
10 // State storage signal declarations
                                                                                                                  10 wire [2:0] ucp3_rd_off;
11 wire dx_clip_space_def;
                                                                                                                 11 wire nxt_ucp3_write_after_cpy;
12
    wire agrd_dx_clip_space_def;
                                                                                                                 12 reg ucp3_write_after_cpy;
13
                                                                                                                 13 wire [2:0] nxt_ucp3_write_ptr;
14
      wire [2:0] ucp0_rd_off;
                                                                                                                 14 reg [2:0] ucp3_write_ptr;
15 wire nxt ucp0 write after cpv;
                                                                                                                 15 wire [2:0] agrd ucp3 write ptr;
16 reg ucp0 write after cpv;
17 wire [2:0] nxt_ucp0_write_ptr;
                                                                                                                 17 wire [2:0] ucp4_rd_off;
18 reg [2:0] ucp0_write_ptr;
                                                                                                                 18 wire nxt_ucp4_write_after_cpy;
19 wire [2:0] agrd_ucp0_write_ptr;
                                                                                                                 19 reg ucp4_write_after_cpy;
                                                                                                                 20 wire [2:0] nxt_ucp4_write_ptr;
21 wire [2:0] ucp1_rd_off;
                                                                                                                 21 reg [2:0] ucp4 write ptr;
22 wire nxt_ucpl_write_after_cpy;
                                                                                                                 22 wire [2:0] agrd_ucp4_write_ptr;
23 reg ucp1_write_after_cpy;
                                                                                                                 23
24 wire [2:0] nxt_ucp1_write_ptr;
                                                                                                                 24 wire [2:0] ucp5_rd_off;
25 reg [2:0] ucp1_write_ptr;
                                                                                                                 25 wire nxt_ucp5_write_after_cpy;
                                         Page 13 of 123
                                                                                                                                                          Page 14 of 123
                                                                        Ex. 2115 - pa_ag.v
                                                                                                                                                                                         Ex. 2115 - pa_ag.v
 1 reg ucp5_write_after_cpy;
                                                                                                                   1 reg [1:0] ccg_ve_b_memsel;
                                                                                                                  2 reg [2:0] ccg_ve_ax_select;
 2 wire [2:0] nxt_ucp5_write_ptr;
 3 reg [2:0] ucp5_write_ptr;
                                                                                                                      reg [2:0] ccg_ve_ay_select;
      wire [2:0] agrd ucp5 write ptr;
                                                                                                                   4 reg [2:0] ccg_ve_az_select;
                                                                                                                   5 reg [2:0] ccg_ve_aw_select;
 6 wire [2:0] gb_rd_off;
                                                                                                                   6 reg [2:0] ccg_ve_bx_select;
 7 wire nxt_gb_write_after_cpy;
                                                                                                                   7 reg [2:0] ccg_ve_by_select;
 8 reg gb_write_after_cpy;
                                                                                                                   8 reg [2:0] ccg_ve_bz_select;
      wire [2:0] nxt_gb_write_ptr;
                                                                                                                      reg [2:0] ccg_ve_bw_select;
10 reg [2:0] gb_write_ptr;
                                                                                                                  10 reg ccg_ve_a_is_www;
11
      wire [2:0] agrd_gb_write_ptr;
                                                                                                                 11 reg ccg_ve_broadcast_x;
12
                                                                                                                  12 reg ccg_ve_abs_a;
      wire [2:0] pntsz_rd_off;
                                                                                                                 13 reg ccg_ve_abs_b;
14 wire nxt_pntsz_write_after_cpy;
                                                                                                                 14 reg ccg_ve_abs_c;
                                                                                                                 15 reg ccg_ve_ax_negate;
15 reg pntsz_write_after_cpy;
      wire [2:0] nxt_pntsz_write_ptr;
                                                                                                                  16 reg ccg_ve_ay_negate;
17 reg [2:0] pntsz_write_ptr;
                                                                                                                  17 reg ccg_ve_az_negate;
18 wire [2:0] agrd_pntsz_write_ptr;
                                                                                                                  18 reg ccg_ve_aw_negate;
                                                                                                                  19 reg ccg_ve_bx_negate;
20 //CCG decode results
                                                                                                                 20 reg ccg_ve_by_negate;
21 reg [2:0] ccg vte opcode;
                                                                                                                 21 reg ccg ve bz negate;
22 reg [2:0] ccg_vte_st_indx;
                                                                                                                 22 reg ccg_ve_bw_negate;
23
                                                                                                                 23 reg ccg_ve_cx_negate;
24 reg [2:0] ccg_ve_opcode;
                                                                                                                 24 reg
                                                                                                                              ccg_ve_cy_negate;
25 reg [1:0] ccg_ve_a_memsel;
                                                                                                                 25 reg
                                                                                                                              ccg_ve_cz_negate;
                                         Page 15 of 123
                                                                                                                                                          Page 16 of 123
                                                                        Ex. 2115 - pa_ag.v
                                                                                                                                                                                         Ex. 2115 - pa_ag.v
```

```
1 reg ccg_ve_cw_negate;
 2 reg ccg_ve_bcc_flat_tst;
                                                                                                           2 reg [1:0] get_clipdist_a_memsel;
 3 reg [2:0] ccg_ve_out_mem_sel;
                                                                                                           3 reg [2:0] get_clipdist_a_select;
 4 reg [5:0] ccg_ve_out_addr;
 5 reg [3:0] ccg_ve_out_we;
                                                                                                           5 //CLIPPER decode results
 6 reg ccg_ve_accum_sel;
                                                                                                           6 reg [2:0] clip_vte_opcode;
     reg [3:0] ccg_ve_pre_acc_we;
                                                                                                           7 reg [2:0] clip_vte_st_indx;
    reg [5:0] ccg_agve_dly_vertex_store_indx;
                                                                                                           9 reg [2:0] clip_ve_opcode;
10 reg [1:0] ccg_agve_dly_valid_bit_set;
                                                                                                          10 reg [1:0] clip_ve_a_memsel;
11 reg [3:0] ccg_agve_dly_user_clip_indx;
                                                                                                          11 reg [1:0] clip_ve_b_memsel;
12 reg ccg_agve_dly_vv_cc_test;
                                                                                                          12 reg [2:0] clip_ve_ax_select;
13 reg ccg_agve_dly_ucp_cc_test;
                                                                                                          13 reg [2:0] clip_ve_ay_select;
14 reg ccg_agve_dly_bcc_cc_test;
                                                                                                          14 reg [2:0] clip_ve_az_select;
15 reg ccg agve dly ps ucp cc test;
                                                                                                          15 reg [2:0] clip ve aw select;
16 reg ccg agve dly ps engh test;
                                                                                                          16 reg [2:0] clip ve bx select;
17
                                                                                                          17 reg [2:0] clip_ve_by_select;
18 reg [1:0] get_baryc_a_memsel;
                                                                                                          18 reg [2:0] clip_ve_bz_select;
19 reg [2:0] get baryc ax select;
                                                                                                          19 reg [2:0] clip_ve_bw_select;
20 reg [2:0] get baryc ay select;
                                                                                                          20 reg clip_ve_a_is_www;
21 reg [2:0] get_baryc_az_select;
                                                                                                          21 reg clip_ve_broadcast_x;
22 reg get_baryc_ax_negate;
                                                                                                          22 reg clip_ve_abs_a;
23 reg
           get_baryc_cx_negate;
                                                                                                          23 reg clip_ve_abs_b;
24
                                                                                                          24 reg
                                                                                                                      clip_ve_abs_c;
25 reg [1:0] get_pos_a_memsel;
                                                                                                                      clip_ve_ax_negate;
                                       Page 17 of 123
                                                                                                                                                 Page 18 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                              Ex. 2115 - pa_ag.v
 l reg clip ve ay negate;
                                                                                                           l reg clip agve dly ps engh test;
 2 reg
           clip_ve_az_negate;
 3 reg
            clip_ve_aw_negate;
 4 reg
            clip ve bx negate;
 5 reg
           clip_ve_by_negate;
                                                                                                           5 // arbiter mux of control
 6 гед
           clip_ve_bz_negate;
                                                                                                           6 reg [2:0] arbsel_vte_opcode;
 7 reg
           clip_ve_bw_negate;
                                                                                                           7 reg [2:0] arbsel_vte_st_indx;
 8 reg
           clip_ve_cx_negate;
                                                                                                           8 reg [2:0] arbsel_ve_opcode;
     reg
           clip_ve_cy_negate;
                                                                                                           9 reg [1:0] arbsel_ve_a_memsel;
10 reg
           clip_ve_cz_negate;
                                                                                                           10 reg [1:0] arbsel_ve_b_memsel;
11 reg
          clip_ve_cw_negate;
                                                                                                          11 reg [2:0] arbsel_ve_ax_select;
12 reg
          clip_ve_bcc_flat_tst;
                                                                                                          12 reg [2:0] arbsel_ve_ay_select;
13 reg [2:0] clip_ve_out_mem_sel;
                                                                                                          13 reg [2:0] arbsel_ve_az_select;
14 reg [5:0] clip_ve_out_addr;
                                                                                                          14 reg [2:0] arbsel_ve_aw_select;
15 reg [3:0] clip_ve_out_we;
                                                                                                          15 reg [2:0] arbsel_ve_bx_select;
16 reg clip_ve_accum_sel;
                                                                                                           16 reg [2:0] arbsel_ve_by_select;
17 reg [3:0] clip_ve_pre_acc_we;
                                                                                                          17 reg [2:0] arbsel_ve_bz_select;
18
                                                                                                          18 reg [2:0] arbsel_ve_bw_select;
19 reg [5:0] clip_agve_dly_vertex_store_indx;
                                                                                                           19 reg arbsel_ve_a_is_www;
20 reg [1:0] clip agve dly valid bit set;
                                                                                                          20 reg arbsel ve broadcast x;
21 reg [3:0] clip agve dly user clip indx;
                                                                                                          21 reg arbsel ve abs a;
22 reg clip_agve_dly_vv_cc_test;
                                                                                                          22 reg arbsel_ve_abs_b;
23 reg clip_agve_dly_ucp_cc_test;
                                                                                                          23 reg arbsel_ve_abs_c;
24 reg clip_agve_dly_bcc_cc_test;
                                                                                                          24 reg arbsel_ve_ax_negate;
25 reg clip_agve_dly_ps_ucp_cc_test;
                                                                                                          25 reg arbsel_ve_ay_negate;
                                      Page 19 of 123
                                                                                                                                                Page 20 of 123
                                                                                                                                                                              Ex. 2115 - pa_ag.v
                                                                   Ex. 2115 - pa_ag.v
```

```
l reg
           arbsel_ve_az_negate;
 2 reg
           arbsel_ve_aw_negate;
                                                                                                             2 reg agve_valid_op;
 3 reg
            arbsel_ve_bx_negate;
 4 reg
            arbsel_ve_by_negate;
                                                                                                             4 //Declare AG_R0 register items
 5 reg
           arbsel_ve_bz_negate;
                                                                                                              5 reg ccg_xfc_r0;
            arbsel_ve_bw_negate;
                                                                                                              6 reg ccg_ve_cc_valid_r0;
 7 reg
           arbsel_ve_cx_negate;
                                                                                                              7 reg [1:0] ccg_sm_state_indx_r0;
                                                                                                              8 reg [2:0] ccg_state_var_indx_r0;
           arbsel_ve_cy_negate;
 9 reg
          arbsel_ve_cz_negate;
                                                                                                                reg [5:0] ccg_vertex_store_indx_r0;
10 reg
          arbsel_ve_cw_negate;
                                                                                                                reg [2:0] ccg_ve_ucp_indx_r0;
11 reg arbsel_ve_bcc_flat_tst;
12 reg [2:0] arbsel_ve_out_mem_sel;
                                                                                                            12 reg clip_xfc_r0;
13 reg [5:0] arbsel_ve_out_addr;
                                                                                                            13 reg [3:0] clip_plane_indx_r0;
14 reg [3:0] arbsel_ve_out_we;
                                                                                                            14 reg [5:0] clip_dst_vertex_indx_r0;
15 reg arbsel ve accum sel;
                                                                                                            15 reg [6:0] clip src vertex indx r0;
16 reg [3:0] arbsel ve pre acc we;
                                                                                                             16 reg clip src vertex type r0;
17
                                                                                                            17 reg clip_ve_ucp_valid_r0;
18 reg [5:0] arbsel agve dly vertex store indx:
                                                                                                            18 reg [6:0] clip_sm_state_indx_r0;
19 reg [1:0] arbsel agve dly valid bit set;
                                                                                                            19 reg [2:0] clip_state_var_indx_r0;
20 reg [3:0] arbsel_agve_dly_user_clip_indx;
21 reg arbsel_agve_dly_vv_cc_test;
                                                                                                            21 //Declare AG R1 register items
22 reg
           arbsel_agve_dly_ucp_cc_test;
                                                                                                            22 reg ccg xfc r1;
23 reg
           arbsel_agve_dly_bcc_cc_test;
                                                                                                            23 //CCG R1 Delay
24 reg
           arbsel_agve_dly_ps_ucp_cc_test;
                                                                                                            24 reg [2:0] ccg_vte_opcode_r1;
25 reg arbsel_agve_dly_ps_engh_test;
                                                                                                            25 reg [2:0] ccg_vte_st_indx_r1;
                                       Page 21 of 123
                                                                                                                                                    Page 22 of 123
                                                                     Ex. 2115 - pa_ag.v
                                                                                                                                                                                 Ex. 2115 - pa_ag.v
 1 reg [2:0] ccg_ve_opcode_r1;
                                                                                                              l reg ccg ve cy negate rl;
 2 reg [1:0] ccg_ve_a_memsel_rl;
                                                                                                             2 reg ccg_ve_cz_negate_rl;
 3 reg [1:0] ccg_ve_b_memsel_rl;
                                                                                                              3 reg ccg_ve_cw_negate_rl;
 4 reg [2:0] ccg_ve_ax_select_rl;
                                                                                                                        ccg_ve_bcc_flat_tst_rl;
 5 reg [2:0] ccg_ve_ay_select_r1;
                                                                                                              5 reg [2:0] ccg_ve_out_mem_sel_rl;
 6 \hspace{0.5cm} \text{reg} \hspace{0.2cm} \textbf{[2:0]} \hspace{0.1cm} \text{ccg\_ve\_az\_select\_r1};
                                                                                                              6 reg [5:0] ccg_ve_out_addr_r1;
 7 reg [2:0] ccg_ve_aw_select_rl;
                                                                                                             7 reg [3:0] ccg_ve_out_we_rl;
 8 reg [2:0] ccg_ve_bx_select_rl;
                                                                                                             8 reg ccg_ve_accum_sel_rl;
 9 reg [2:0] ccg_ve_by_select_rl;
                                                                                                                reg [3:0] ccg_ve_pre_acc_we_rl;
10 reg [2:0] ccg_ve_bz_select_r1;
11 reg [2:0] ccg_ve_bw_select_rl;
                                                                                                            11 reg [5:0] ccg_agve_dly_vertex_store_indx_r1;
12 reg ccg_ve_a_is_wwww_rl;
                                                                                                            12 reg [1:0] ccg_agve_dly_valid_bit_set_rl;
                                                                                                            13 reg [3:0] ccg_agve_dly_user_clip_indx_rl;
13 reg
          ccg_ve_broadcast_x_r1;
14 reg
          ccg_ve_abs_a_rl;
                                                                                                             14 reg ccg_agve_dly_vv_cc_test_rl;
15 reg ccg_ve_abs_b_r1;
                                                                                                             15 reg ccg_agve_dly_ucp_cc_test_rl;
16 reg
           ccg_ve_abs_c_rl;
                                                                                                             16 reg ccg_agve_dly_bcc_cc_test_rl;
17 reg
                                                                                                            17 reg ccg_agve_dly_ps_ucp_cc_test_r1;
          ccg_ve_ax_negate_rl;
18 reg
                                                                                                            18 reg ccg_agve_dly_ps_engh_test_rl;
           ccg ve ay negate r1;
19 гед
            ccg ve az negate rl;
20 reg
                                                                                                            20 reg clip xfc r1;
           ccg ve aw negate rl;
                                                                                                            21 //CLIPPER R1 Delay
21 reg
           ccg ve bx negate rl;
22 reg
                                                                                                            22 reg [2:0] clip vte opcode r1;
            ccg ve by negate rl;
23 reg
                                                                                                            23 reg [2:0] clip_vte_st_indx_r1;
            ccg ve bz negate r1;
24 reg
                                                                                                            24 reg [2:0] clip_ve_opcode_r1;
            ccg ve bw negate rl;
25 reg
           ccg ve cx negate rl;
                                                                                                            25 reg [1:0] clip_ve_a_memsel_r1;
                                       Page 23 of 123
                                                                                                                                                    Page 24 of 123
                                                                    Ex. 2115 - pa_ag.v
                                                                                                                                                                                 Ex. 2115 - pa_ag.v
```

```
l reg [1:0] clip_ve_b_memsel_rl;
                                                                                                           l reg clip_ve_cw_negate_rl;
 2 reg [2:0] clip_ve_ax_select_r1;
                                                                                                           2 reg clip_ve_bcc_flat_tst_rl;
 3 reg [2:0] clip_ve_ay_select_r1;
                                                                                                           3 reg [2:0] clip_ve_out_mem_sel_r1;
 4 reg [2:0] clip_ve_az_select_rl;
                                                                                                           4 reg [5:0] clip_ve_out_addr_r1;
 5 reg [2:0] clip_ve_aw_select_rl;
                                                                                                           5 reg [3:0] clip_ve_out_we_r1;
 6 reg [2:0] clip_ve_bx_select_r1;
                                                                                                           6 reg clip_ve_accum_sel_rl;
 7 reg [2:0] clip_ve_by_select_r1;
                                                                                                           7 reg [3:0] clip_ve_pre_acc_we_r1;
 8 reg [2:0] clip_ve_bz_select_r1;
     reg [2:0] clip_ve_bw_select_r1;
                                                                                                              reg [5:0] clip_agve_dly_vertex_store_indx_rl;
10 reg
          clip_ve_a_is_wwww_rl;
                                                                                                          10 reg [1:0] clip_agve_dly_valid_bit_set_r1;
11 reg
          clip_ve_broadcast_x_r1;
                                                                                                          11 reg [3:0] clip_agve_dly_user_clip_indx_r1;
12 reg
          clip_ve_abs_a_r1;
                                                                                                          12 reg clip_agve_dly_vv_cc_test_r1;
13 reg
           clip_ve_abs_b_rl;
                                                                                                          13 reg clip_agve_dly_ucp_cc_test_rl;
14 reg
           clip_ve_abs_c_rl;
                                                                                                          14 reg clip_agve_dly_bcc_cc_test_r1;
15 гед
                                                                                                          15 reg clip agve dly ps ucp cc test r1;
           clip ve ax negate rl;
16 гед
                                                                                                          16 reg clip agve dly ps engh test r1;
           clip ve ay negate rl;
17 reg
                                                                                                          17
            clip ve az negate r1;
18 reg
                                                                                                          18 //Declare AG R1 register items
            clip ve aw negate r1;
19 reg
            clip_ve_bx_negate_r1;
                                                                                                          19 reg [2:0] vte_opcode_r2;
                                                                                                          20 reg [2:0] vte_st_indx_r2;
20 reg
            clip_ve_by_negate_r1;
21 reg
            clip_ve_bz_negate_rl;
                                                                                                          21 reg [2:0] ve opcode r2;
22 reg
            clip_ve_bw_negate_rl;
                                                                                                          22 reg [1:0] ve a memsel r2;
23 reg
            clip_ve_cx_negate_r1;
                                                                                                          23 reg [1:0] ve_b_memsel_r2;
24 reg
            clip_ve_cy_negate_rl;
                                                                                                          24 reg [2:0] ve_ax_select_r2;
25 reg
            clip_ve_cz_negate_rl;
                                                                                                          25 reg [2:0] ve_ay_select_r2;
                                      Page 25 of 123
                                                                                                                                                Page 26 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                             Ex. 2115 - pa_ag.v
 l reg [2:0] ve_az_select_r2;
                                                                                                           1 reg [5:0] ve_out_addr_r2;
 2 reg [2:0] ve_aw_select_r2;
                                                                                                           2 reg [3:0] ve out we r2;
 3 reg [2:0] ve bx select r2;
                                                                                                           3 reg ve accum sel r2;
 4 reg [2:0] ve by select r2;
                                                                                                           4 reg [3:0] ve_pre_acc_we_r2;
 5 reg [2:0] ve_bz_select_r2;
 6 reg [2:0] ve_bw_select_r2;
                                                                                                           6 reg [5:0] agve_dly_vertex_store_indx_r2;
 7 reg ve_a_is_wwww_r2;
                                                                                                           7 reg [1:0] agve_dly_valid_bit_set_r2;
 8 reg
           ve broadcast x r2;
                                                                                                           8 reg [3:0] agve_dly_user_clip_indx_r2;
     reg
           ve_abs_a_r2;
                                                                                                           9 reg agve_dly_vv_cc_test_r2;
           ve_abs_b_r2;
10 reg
                                                                                                          10 reg agve_dly_ucp_cc_test_r2;
11 reg
           ve_abs_c_r2;
                                                                                                          11 reg agve_dly_bcc_cc_test_r2;
12 reg
           ve_ax_negate_r2;
                                                                                                          12 reg agve_dly_ps_ucp_cc_test_r2;
13 reg
                                                                                                          13 reg agve_dly_ps_engh_test_r2;
           ve_ay_negate_r2;
14 reg
                                                                                                          14
          ve_az_negate_r2;
15 reg
                                                                                                          15 //Declare AG_R3 output register items
           ve_aw_negate_r2;
16 reg
           ve_bx_negate_r2;
17 reg
                                                                                                          17 reg [2:0] ag_vte_opcode;
           ve_by_negate_r2;
18 reg
            ve bz negate r2;
                                                                                                          18 reg [2:0] ag_vte_st_indx;
19 reg
                                                                                                          19 reg [1:0] ag_vte_vertex_store_indx;
            ve bw negate r2;
20 reg
                                                                                                          20 reg [2:0] ag ve opcode;
           ve cx negate r2;
21 reg
                                                                                                          21 reg [31:0] ag ve in a0;
           ve cv negate r2;
                                                                                                          22 reg [31:0] ag_ve_in_a1;
22 reg
            ve cz negate r2;
23 reg
                                                                                                          23 reg [31:0] ag_ve_in_a2;
            ve cw negate r2;
24 reg
                                                                                                          24 reg [31:0] ag ve in a3;
           ve bcc flat tst r2;
25 reg [2:0] ve_out_mem_sel_r2;
                                                                                                          25 reg [31:0] ag ve in b0;
                                      Page 27 of 123
                                                                                                                                                Page 28 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                             Ex. 2115 - pa_ag.v
```

```
1 reg [31:0] ag ve in b1;
                                                                                                           l reg [3:0] ag_ve_pre_accum_we;
 2 reg [31:0] ag_ve_in_b2;
 3 reg [31:0] ag_ve_in_b3;
                                                                                                          3 reg [127:0] amem sel data;
         ag_ve_a_is_wwww;
                                                                                                          4 reg [127:0] bmem_sel_data;
          ag_ve_broadcast_x;
                                                                                                           5 wire [31:0] agswz_ve_in_a0;
           ag_ve_abs_a;
                                                                                                           6 wire [31:0] agswz_ve_in_al;
                                                                                                           7 wire [31:0] agswz_ve_in_a2;
           ag_ve_abs_b;
                                                                                                           8 wire [31:0] agswz_ve_in_a3;
           ag_ve_abs_c;
 9 reg
                                                                                                           9 wire [31:0] agswz_ve_in_b0;
          ag_ve_ax_negate;
10 reg
                                                                                                          10 wire [31:0] agswz_ve_in_b1;
           ag_ve_ay_negate;
11 reg ag_ve_az_negate;
                                                                                                         11 wire [31:0] agswz_ve_in_b2;
12 reg
         ag_ve_aw_negate;
                                                                                                         12 wire [31:0] agswz_ve_in_b3;
13 reg
           ag ve bx negate;
14 reg
                                                                                                         14 reg agve_dly_valid_op_r3;
           ag ve by negate;
15 reg
                                                                                                         15 reg [5:0] agve dly vertex store indx r3;
           ag ve bz negate;
16 гед
                                                                                                          16 reg [1:0] agve dly valid bit set r3;
           ag ve bw negate;
17 reg
                                                                                                          17 reg [3:0] agve_dly_user_clip_indx_r3;
            ag ve cx negate;
18 reg
                                                                                                         18 reg agve_dly_vv_cc_test_r3;
            ag ve cy negate;
19 reg
            ag ve cz negate;
                                                                                                         19 reg agve_dly_ucp_cc_test_r3;
20 reg
            ag ve cw negate;
                                                                                                         20 reg agve_dly_bcc_cc_test_r3;
21 reg
            ag ve bcc flat tst;
                                                                                                         21 reg agve_dly_ps_ucp_cc_test_r3;
22 reg [2:0] ag_ve_out_mem_sel;
                                                                                                         22 reg agve_dly_ps_engh_test_r3;
23 reg [5:0] ag_ve_out_addr;
                                                                                                         23
24 reg [3:0] ag_ve_out_we;
                                                                                                         24 //delay pipe
25 reg ag_ve_accum_sel;
                                                                                                         25 reg [17:0] agve_dly0;
                                      Page 29 of 123
                                                                                                                                                Page 30 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                            Ex. 2115 - pa_ag.v
 1 reg [17:0] agve_dly1;
                                                                                                                 ccg ve valid,
 2 reg [17:0] agve dly2:
                                                                                                                 ccg sm state indx,
     reg [17:0] agve dly3;
                                                                                                                 ccg state var indx,
 4 reg [17:0] agve dly4;
                                                                                                                 ccg vertex store indx,
 5 reg [17:0] agve_dly5;
                                                                                                                 ccg_ve_ucp_indx} = ccg_to_arb_data;
 7 reg agve_dly_valid_op;
                                                                                                                 \{clip\_plane\_indx,
 8 reg [5:0] agve_dly_vertex_store_indx;
                                                                                                                 clip dst vertex indx,
     reg [1:0] agve_dly_valid_bit_set;
                                                                                                                 clip_src_vertex_indx,
10 reg [3:0] agve_dly_user_clip_indx;
                                                                                                         10
                                                                                                                 clip_src_vertex_type,
11 reg agve_dly_vv_cc_test;
                                                                                                         11
                                                                                                                 clip_ve_ucp_valid,
12 reg agve_dly_ucp_cc_test;
                                                                                                         12
                                                                                                                 clip_sm_state_indx,
13 reg agve_dly_bcc_cc_test;
                                                                                                                 clip_state_var_indx} = clip_to_arb_data;
14 reg
                                                                                                         14 end
         agve_dly_ps_ucp_cc_test;
15 reg
                                                                                                         15
         agve_dly_ps_engh_test;
17 assign nxt_ucp0_write_after_cpy = dirty(srst, rbiu_cpy, rbiu_ag_ucp0_sel);
18 // Combinational logic
                                                                                                         18 assign nxt ucpl write after cpy = dirty(srst, rbiu cpy, rbiu ag ucpl sel);
     19 assign nxt_ucp2_write_after_cpy = dirty(srst, rbiu_cpy, rbiu_ag_ucp2_sel);
                                                                                                         20 assign nxt ucp3 write after cpy = dirty(srst, rbiu cpy, rbiu ag ucp3 sel);
21 //map ccg and clip controller signals to broken out names
                                                                                                         21 assign nxt ucp4 write after cpy = dirty(srst, rbiu cpy, rbiu ag ucp4 sel);
22 always @(ccg_to_arb_data or
                                                                                                         22 assign nxt_ucp5_write_after_cpy = dirty(srst, rbiu_cpy, rbiu_ag_ucp5_sel);
23
                                                                                                         23 \quad \  \  assign\ nxt\_gb\_write\_after\_cpy \quad = dirty(srst, rbiu\_cpy, rbiu\_ag\_gb\_sel);
         clip_to_arb_data)
24 begin
                                                                                                         24 assign nxt_pntsz_write_after_cpy = dirty(srst, rbiu_cpy, rbiu_ag_pntsz_sel);
      {ccg_ve_cc_valid,
                                                                                                         2.5
                                      Page 31 of 123
                                                                                                                                               Page 32 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                            Ex. 2115 - pa_ag.v
```

```
1 //maintain state next write ptr's
                                                                                                                                 end if (rbiu_ag_ucp1_sel)begin
 2 \quad assign\ nxt\_ucp0\_write\_ptr = stwtptr(srst, rbiu\_cpy, ucp0\_write\_after\_cpy,\ ucp0\_write\_ptr);
                                                                                                                                   stve_wa = {3'b001, ucp1_write_ptr };
 3 \quad assign \ nxt\_ucp1\_write\_ptr = stwtptr(srst, rbiu\_cpy, ucp1\_write\_after\_cpy, \ ucp1\_write\_ptr);
                                                                                                                                   stve_we = rbiu_ag_ucpl_sel;
      assign nxt_ucp2_write_ptr = stwtptr(srst, rbiu_cpy, ucp2_write_after_cpy, ucp2_write_ptr);
                                                                                                                                 end if (rbiu_ag_ucp2_sel)begin
      assign nxt_ucp3_write_ptr = stwtptr(srst, rbiu_cpy, ucp3_write_after_cpy, ucp3_write_ptr);
                                                                                                                                   stve_wa = {3'b010, ucp2_write_ptr };
      assign\ nxt\_ucp4\_write\_ptr\ = stwtptr(srst, rbiu\_cpy, ucp4\_write\_after\_cpy,\ ucp4\_write\_ptr);
                                                                                                                                   stve_we = rbiu_ag_ucp2_sel;
      assign nxt_ucp5_write_ptr = stwtptr(srst, rbiu_cpy, ucp5_write_after_cpy, ucp5_write_ptr);
                                                                                                                                 end if (rbiu_ag_ucp3_sel)begin
      assign nxt_gb_write_ptr = stwtptr(srst, rbiu_cpy, gb_write_after_cpy, _gb_write_ptr);
                                                                                                                                   stve_wa = {3'b011, ucp3_write_ptr };
      assign nxt_pntsz_write_ptr = stwtptr(srst, rbiu_cpy, pntsz_write_after_cpy, pntsz_write_ptr);
                                                                                                                                   stve_we = rbiu_ag_ucp3_sel;
                                                                                                                                 end if (rbiu_ag_ucp4_sel)begin
11
     //select and assemble state mem write address
                                                                                                                                  stve_wa = {3'b100, ucp4_write_ptr };
12
    always @(rbiu_ag_ucp0_sel or ucp0_write_ptr or
                                                                                                                                   stve_we = rbiu_ag_ucp4_sel;
13
           rbiu ag ucpl sel or ucpl write ptr or
                                                                                                                                 end if (rbiu ag ucp5 sel)begin
14
           rbiu ag ucp2 sel or ucp2 write ptr or
                                                                                                                                  stve wa = {3'b101, ucp5 write ptr };
15
           rbiu ag ucp3 sel or ucp3 write ptr or
                                                                                                                         15
                                                                                                                                  stve we = rbiu ag ucp5 sel;
16
           rbiu ag ucp4 sel or ucp4 write ptr or
                                                                                                                                 end if (rbiu ag gb sel)begin
17
                                                                                                                                  stve wa = {3'b110, gb_write_ptr };
           rbiu ag ucp5 sel or ucp5 write ptr or
                                                                                                                         17
18
           rbiu_ag_gb_sel or _gb_write_ptr or
                                                                                                                         18
                                                                                                                                  stve we = rbiu ag gb sel;
19
           rbiu_ag_pntsz_sel or pntsz_write_ptr)
                                                                                                                         19
                                                                                                                                 end if (rbiu_ag_pntsz_sel)begin
20 begin
                                                                                                                         20
                                                                                                                                  stve wa = {3'b111, pntsz write ptr };
21
         stve wa = 6'b0000000:
                                                                                                                         21
                                                                                                                                   stve_we = rbiu_ag_pntsz_sel;
22
         stve we = 4'b0000:
                                                                                                                         22
23
         if (rbiu_ag_ucp0_sel) begin
                                                                                                                         23
24
          stve_wa = {3'b000, ucp0_write_ptr };
                                                                                                                         24
                                                                                                                         25 //Arbiter logic
25
          stve_we = rbiu_ag_ucp0_sel;
                                           Page 33 of 123
                                                                                                                                                                     Page 34 of 123
                                                                             Ex. 2115 - pa_ag.v
                                                                                                                                                                                                      Ex. 2115 - pa_ag.v
 l assign {arb_clip_xfc, arb_ccg_xfc} = arb(srst, clip_ve_valid, ccg_ve_valid);
                                                                                                                                     nxt_state_type[5:3] = PNTSZ;
 2
                                                                                                                                   end else if (clip_sm_state_indx == SMC_CLIP_DIST_VV) begin
 3 //Arbiter generation of intermediate stye raddr
                                                                                                                                     nxt_stve_re = clip_ve_valid;
 4 always @(clip ve valid or
                                                                                                                                     nxt state type[5:3] = GB;
           clip_state_var_indx or
                                                                                                                                   end else if ((clip_sm_state_indx == SMC_CLIP_DIST_UCP) ||
           clip plane indx or
                                                                                                                                          (clip_sm_state_indx == SMC_PS_CLIP_DIST_UCP) ||
           clip_sm_state_indx or
                                                                                                                                           (clip\_sm\_state\_indx == SMC\_PS\_UCP\_DIST\_UL) \parallel
           ccg_ve_valid or
                                                                                                                                           (clip sm state indx == SMC PS UCP DIST UR) ||
           ccg_state_var_indx or
                                                                                                                                           (clip\_sm\_state\_indx == SMC\_PS\_UCP\_DIST\_LR) \parallel
                                                                                                                                           (clip_sm_state_indx == SMC_PS_UCP_DIST_LL)) begin
10
           ccg_ve_ucp_indx or
                                                                                                                         10
11
           ccg sm state indx)
                                                                                                                         11
                                                                                                                                     nxt\_stve\_re = clip\_ve\_valid;
                                                                                                                                     nxt\_state\_type[5:3] = clip\_plane\_indx;
12
                                                                                                                         12
13
                                                                                                                         13
         nxt_stve_re = 1'b0;
14
         nxt_state_type[5:3] = 3'b000;
                                                                                                                         14
                                                                                                                                  end else begin //ccg wins arbitration
15
         nxt_arb_state_var_indx = 3'b000;
                                                                                                                                   nxt_arb_state_var_indx = ccg_state_var_indx;
16
         arb_ucp_indx = 3'b000;
                                                                                                                                   arb\_ucp\_indx \hspace{1.5cm} = ccg\_ve\_ucp\_indx; \\
17
                                                                                                                                   if(ccg\_sm\_state\_indx == SMCC\_EMPTY) begin
18
         if (clip_ve_valid == 1'b1) begin
                                                                                                                                    nxt_stve_re = ccg_ve_valid;
19
                                                                                                                                   nxt_state_type[5:3] = GB; //guard band lookup
          nxt arb state var indx = clip state var indx;
20
          arb ucp indx = clip plane indx;
                                                                                                                         20
                                                                                                                                   end else begin
                                                                                                                                    nxt_stve_re = ccg_ve_valid;
21
                                                                                                                         21
22
                                                                                                                         22
                                                                                                                                    nxt_state_type[5:3] = ccg_ve_ucp_indx;
          //need to add clipper read address for stve
23
                                                                                                                         23
          if ((clip sm state indx == SMC PS CULL RADIUS VPORT) ||
24
            (clip sm state indx == SMC PS XY RADIUS VPORT)) begin
                                                                                                                         24
25
            nxt stve re = clip ve valid;
                                                                                                                         25
                                                                                                                                  end
                                           Page 35 of 123
                                                                                                                                                                     Page 36 of 123
                                                                             Ex. 2115 - pa ag.v
                                                                                                                                                                                                      Ex. 2115 - pa ag.v
```

```
1
      end
                                                                                                                1
                                                                                                                         clip_to_ag_point_buf_raddr)
2
                                                                                                                2 begin
3 //determine the final stye_raddr
                                                                                                                3
                                                                                                                      pos_re = ccg_xfc_r0;
 4 always @(state_type_r0 or ucp0_rd_off or ucp1_rd_off or ucp2_rd_off or ucp3_rd_off or
                                                                                                                      pos_raddr = ccg_vertex_store_indx_r0;
 5
                ucp4\_rd\_off\ or\ ucp5\_rd\_off\ or\ ucp1\_rd\_off\ or\ gb\_rd\_off\ or\ pntsz\_rd\_off)
                                                                                                                       pntsz_re = 0;
      begin
 6
                                                                                                                      pntsz_raddr = clip_src_vertex_indx_r0[5:0];
        case (state_type_r0)
        UCP0 : begin stve_raddr = {UCP0, ucp0_rd_off}; end
                                                                                                                      if (ccg_xfc_r0) begin
         UCP1 : begin stve_raddr = {UCP1, ucp1_rd_off}; end
                                                                                                                        pos_re = 1;
         UCP2: begin stve\_raddr = \{UCP2, ucp2\_rd\_off \}; end
                                                                                                                        pos_raddr = ccg_vertex_store_indx_r0;
11
         UCP3 : begin stve_raddr = {UCP3, ucp3_rd_off}; end
                                                                                                               11
                                                                                                                        pntsz_re = 0;
12
         UCP4 : begin stve_raddr = {UCP4, ucp4_rd_off}; end
                                                                                                               12
                                                                                                                        pntsz_raddr = clip_src_vertex_indx_r0[5:0];
13
         UCP5 : begin stve_raddr = {UCP5, ucp5_rd_off}; end
                                                                                                               13
14
         GB : begin stve_raddr = {GB, gb_rd_off }; end
                                                                                                                       else if (clip_xfc_r0) begin
15
         PNTSZ: begin stve raddr = {PNTSZ,pntsz rd off}; end
                                                                                                               15
                                                                                                                        pos re = 1;
16
        endcase
                                                                                                               16
                                                                                                                        pos_raddr = clip_src_vertex_indx_r0[5:0];
17
                                                                                                               17
      end
                                                                                                                        pntsz re = 1:
18
                                                                                                               18
                                                                                                                        pntsz_raddr = clip_src_vertex_indx_r0[5:0];
19
    //determine the POS and PNTSZ read addr
                                                                                                               19
20 always @(ccg xfc r0 or
                                                                                                               20
                                                                                                                       else if (clip_to_ag_point_buf_re) begin
                                                                                                                        pos_re = 0;
21
         ccg vertex store indx r0 or
                                                                                                               21
22
         clip xfc r0 or
                                                                                                               22
                                                                                                                        pos_raddr = clip_src_vertex_indx_r0[5:0];
23
         clip_sm_state_indx_r0 or
                                                                                                               23
24
         clip_src_vertex_indx_r0 or
                                                                                                               24
                                                                                                                        pntsz_raddr = clip_to_ag_point_buf_raddr;
         clip_to_ag_point_buf_re or
                                                                                                               25
                                        Page 37 of 123
                                                                                                                                                       Page 38 of 123
                                                                      Ex. 2115 - pa_ag.v
                                                                                                                                                                                     Ex. 2115 - pa_ag.v
                                                                                                                                         clip_src_vertex_indx_r0[3:0] };
2
                                                                                                                2
                                                                                                                        end
3 //determine the ve_veoc_vector_back memory read addr
                                                                                                                        SMC OUTPUT FIRST CLIP POS 0,
4 always @(clip_xfc_r0 or
                                                                                                                4
         clip_sm_state_indx_r0 or
                                                                                                                        SMC OUTPUT FIRST CLIP POS 1.
                                                                                                                        SMC_OUTPUT_FIRST_CLIP_POS_2,
         clip_src_vertex_indx_r0 or
                                                                                                                        SMC_OUTPUT_REST_CLIP_POS,
          clip_dst_vertex_indx_r0 or
                                                                                                                        SMC_T_BLEND_PREV_POS_0,
 8
          clip_src_vertex_type_r0)
 9
                                                                                                                        SMC_T_BLEND_CURR_POS_0,
10
        ve\_veoc\_vector\_back\_re = clip\_xfc\_r0;
                                                                                                               10
                                                                                                                        SMC_T_BLEND_PREV_POS_1,
11
        ve_veoc_vector_back_raddr = 5'b000000;
                                                                                                               11
                                                                                                                        SMC_T_BLEND_CURR_POS_1,
12
                                                                                                               12
                                                                                                                        SMC_CLIP_DIST_VV,
13
                                                                                                                        SMC_CLIP_DIST_UCP : begin
        case (clip_sm_state_indx_r0)
14
                                                                                                               14
                                                                                                                         ve_veoc_vector_back_re = clip_xfc_r0;
15
         SMC_OUTPUT_FIRST_BARYC_0,
                                                                                                                         // GetPosition subroutine
16
         SMC_OUTPUT_FIRST_BARYC_1,
                                                                                                                         if(clip_src_vertex_type_r0==1'b1) begin
17
         SMC_OUTPUT_FIRST_BARYC_2,
                                                                                                                          ve_veoc_vector_back_raddr = { 2'b01,
18
         SMC_OUTPUT_REST_BARYC,
                                                                                                                                           clip src vertex indx r0[3:0] };
19
         SMC_T_BLEND_PREV_ABC_0,
                                                                                                                         end else begin
         SMC_T_BLEND_CURR_ABC_0,
20
                                                                                                               20
                                                                                                                          case (clip src vertex indx r0[1:0])
21
         SMC T BLEND PREV ABC 1,
                                                                                                               21
                                                                                                                          PS VERT UL : begin
                                                                                                                            ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_UL;
22
         SMC_T_BLEND_CURR_ABC_1 : begin
                                                                                                               22
23
          ve_veoc_vector_back_re = clip_xfc_r0;
                                                                                                               23
24
                                                                                                               24
          // GetBarycCoord subroutine
                                                                                                                           PS VERT UR : begin
25
           ve veoc vector back raddr = { 2'b00,
                                                                                                               25
                                                                                                                             ve veoc vector back raddr = VEOC PS POS VERT UR;
                                        Page 39 of 123
                                                                                                                                                       Page 40 of 123
                                                                                                                                                                                     Ex. 2115 - pa_ag.v
                                                                      Ex. 2115 - pa ag.v
```

```
end else begin
           PS_VERT_LL : begin
 2
                                                                                                       2
                                                                                                                 ve_veoc_vector_back_raddr = VEOC_CLIP_DIST_ORIG;
            ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_LL;
                                                                                                       3
                                                                                                       4
                                                                                                                // end GetClipDist
           PS VERT LR: begin
            ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_LR;
                                                                                                               SMC\_PS\_UCP\_DIST\_UL: begin
                                                                                                                ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_UL;
             ve_veoc_vector_back_raddr = 6'bxxxxxx;
10
                                                                                                      10
11
           endcase
                                                                                                      11
                                                                                                               SMC_PS_UCP_DIST_UR : begin
12
         end
                                                                                                      12
                                                                                                               ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_UR;
13
                                                                                                      13
14
                                                                                                      14
        SMC_EDGE_DISTANCE_0,
15
                                                                                                      15
                                                                                                               SMC PS UCP DIST LL: begin
        SMC_EDGE_DISTANCE_1,
                                                                                                               ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_LL;
16
                                                                                                      16
17
        SMC CLIP T FACTOR PREV 0.
                                                                                                      17
18
        SMC CLIP T FACTOR CURR 0.
                                                                                                      18
19
        SMC CLIP T FACTOR PREV 1,
                                                                                                      19
                                                                                                               SMC PS UCP DIST LR : begin
20
        SMC CLIP T FACTOR CURR 1: begin
                                                                                                               ve_veoc_vector_back_raddr = VEOC_PS_POS_VERT_LR;
                                                                                                      20
21
         // GetClipDist subroutine
                                                                                                      21
22
         ve veoc vector back re = clip xfc r0;
                                                                                                      22
23
         if(clip_src_vertex_type_r0==1'b1) begin
                                                                                                      23
                                                                                                               SMC PS ENGH TEST: begin
24
          ve_veoc_vector_back_raddr = { 2'b00,
                                                                                                      24
                                                                                                               ve_veoc_vector_back_raddr = VEOC_CLIP_DIST_ORIG;
25
                          clip_src_vertex_indx_r0[3:0] };
                                                                                                      25
                                     Page 41 of 123
                                                                                                                                           Page 42 of 123
                                                                 Ex. 2115 - pa_ag.v
                                                                                                                                                                       Ex. 2115 - pa_ag.v
                                                                                                             ccg_ve_bw_select = SRC_SELECT_FORCE_0;
2
       endcase
                                                                                                             ccg ve a is wwww = 1'b0;
3
      end
                                                                                                             ccg ve broadcast x = 1'b0;
4
                                                                                                             ccg ve abs a = 1'b0;
5 // CCG DECODE
                                                                                                             ccg_ve_abs_b = 1'b0;
 6 always @(ccg_state_var_indx_r0 or
                                                                                                             ccg_ve_abs_c = 1'b0;
        ccg_ve_ucp_indx_r0 or
                                                                                                             ccg_ve_ax_negate = 1'b0;
         ccg_ve_cc_valid_r0 or
                                                                                                             ccg_ve_ay_negate = 1'b0;
         ccg_vertex_store_indx_r0 or
                                                                                                             ccg_ve_az_negate = 1'b0;
10
        ccg_sm_state_indx_r0 or
                                                                                                      10
                                                                                                             ccg_ve_aw_negate = 1'b0;
11
         ccg_xfc_r0 or
                                                                                                      11
                                                                                                             ccg_ve_bx_negate = 1'b0;
                                                                                                            ccg_ve_by_negate = 1'b0;
12
         dx_clip_space_def)
                                                                                                      12
13
                                                                                                            ccg_ve_bz_negate = 1'b0;
14
                                                                                                            ccg_ve_bw_negate = 1'b0;
15
      ccg_vte_opcode = VTE_NO_OP;
                                                                                                           ccg_ve_cx_negate = 1'b0;
16
      ccg_ve_opcode = VECTOR_NO_OP;
                                                                                                            ccg_ve_cy_negate = 1'b0;
17
      ccg_ve_a_memsel = VEA_MEMSEL_VEOC_VE;
                                                                                                            ccg_ve_cz_negate = 1'b0;
18
      ccg_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                                                                                                            ccg ve cw negate = 1'b0;
19
      ccg_ve_ax_select = SRC_SELECT_FORCE_0;
                                                                                                            ccg ve bcc flat tst = 1'b0;
20
      ccg ve av select = SRC SELECT FORCE 0;
                                                                                                      20
                                                                                                            ccg ve out mem sel = 3'b0;
      ccg_ve_az_select = SRC_SELECT_FORCE_0;
                                                                                                           ccg_ve_out_addr = 5'b000000;
21
                                                                                                      21
                                                                                                           ccg_ve_out_we = 4'b0000;
22
      ccg_ve_aw_select = SRC_SELECT_FORCE_0;
                                                                                                      22
23
      ccg_ve_bx_select = SRC_SELECT_FORCE_0;
                                                                                                      23
                                                                                                             ccg_ve_accum_sel = 1'b0;
24
       ccg ve by select = SRC SELECT FORCE 0;
                                                                                                      24
                                                                                                             ccg_ve_pre_acc_we = 4'b0000;
       ccg ve bz select = SRC SELECT FORCE 0;
                                     Page 43 of 123
                                                                                                                                            Page 44 of 123
                                                                                                                                                                       Ex. 2115 - pa_ag.v
                                                                 Ex. 2115 - pa ag.v
```

```
ccg ve a memsel = VEA MEMSEL POS BUF VE;
       ccg_agve_dly_vertex_store_indx = ccg_vertex_store_indx_r0;
2
       ccg_agve_dly_valid_bit_set = 2'b0;
                                                                                                                      ccg ve ax select = SRC SELECT X;
       ccg_agve_dly_user_clip_indx = 4'b0;
                                                                                                                     ccg_ve_ay_select = SRC_SELECT_Y;
       ccg_agve_dly_vv_cc_test = 1'b0;
                                                                                                                     ccg_ve_az_select = SRC_SELECT_Z;
       ccg_agve_dly_ucp_cc_test = 1'b0;
                                                                                                                     ccg_ve_aw_select = SRC_SELECT_W;
       ccg_agve_dly_bcc_cc_test = 1'b0;
                                                                                                                     ccg_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                                                                                                                     ccg_ve_bx_select = SRC_SELECT_X;
       ccg_agve_dly_ps_ucp_cc_test = 1'b0;
       ccg_agve_dly_ps_engh_test = 1'b0;
                                                                                                                     ccg_ve_by_select = SRC_SELECT_Y;
                                                                                                                     ccg_ve_bz_select = SRC_SELECT_Z;
10
                                                                                                                     ccg_ve_bw_select = SRC_SELECT_W;
       ccg_vte_st_indx = ccg_state_var_indx_r0;
11
12
       if (ccg_xfc_r0 == 1'b1) begin
                                                                                                           12
13
                                                                                                           13
                                                                                                                    SMCC_CLIP_CODE_UCP: begin
        case (ccg sm state indx r0)
14
         SMCC_EMPTY : begin
                                                                                                                     ccg_ve_opcode = VE_DOT_PRODUCT;
                                                                                                                     ccg_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
15
          if (dx clip space def) begin
                                                                                                           15
16
           ccg ve opcode = VE CLIP CODE VV Z0W;
                                                                                                           16
                                                                                                                     ccg ve ax select = SRC SELECT X;
17
                                                                                                           17
                                                                                                                     ccg_ve_ay_select = SRC_SELECT_Y;
           end else begin
18
           ccg_ve_opcode = VE_CLIP_CODE_VV_ZWW;
                                                                                                           18
                                                                                                                     ccg ve az select = SRC SELECT Z:
19
                                                                                                           19
                                                                                                                      ccg ve aw select = SRC SELECT W;
           ccg_vte_opcode = VTE ORIG POS:
                                                                                                                     ccg ve b memsel = VEB MEMSEL STVE VE:
20
                                                                                                           20
21
           ccg ve a is wwww = 1'b1;
                                                                                                           21
                                                                                                                     ccg ve bx select = SRC SELECT X;
22
           ccg_agve_dly_vv_cc_test = 1'b1;
                                                                                                           22
                                                                                                                     ccg ve by select = SRC SELECT Y;
23
           if(ccg ve cc valid r0) begin
                                                                                                          23
                                                                                                                     ccg_ve_bz_select = SRC_SELECT_Z;
24
           ccg_agve_dly_valid_bit_set = CC_VALID;
                                                                                                          24
                                                                                                                      ccg_ve_bw_select = SRC_SELECT_W;
25
                                                                                                           25
                                                                                                                      if(ccg_ve_cc_valid_r0) begin
                                      Page 45 of 123
                                                                                                                                                 Page 46 of 123
                                                                    Ex. 2115 - pa_ag.v
                                                                                                                                                                               Ex. 2115 - pa_ag.v
            ccg_agve_dly_valid_bit_set = CC_VALID;
                                                                                                                  clip_ve_a_memsel = VEA_MEMSEL_VEOC_VE;
                                                                                                                  clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
 2
                                                                                                                  clip_ve_ax_select = SRC_SELECT_FORCE_0;
          ccg_agve_dly_user_clip_indx = ccg_ve_ucp_indx_r0;
 4
                                                                                                                  clip_ve_ay_select = SRC_SELECT_FORCE_0;
           ccg_agve_dly_ucp_cc_test = l'b1;
 5
          end
                                                                                                                  clip_ve_az_select = SRC_SELECT_FORCE_0:
                                                                                                                  clip_ve_aw_select = SRC_SELECT_FORCE_0;
                                                                                                                  clip_ve_bx_select = SRC_SELECT_FORCE_0;
          default
                                                                                                                  clip_ve_by_select = SRC_SELECT_FORCE_0;
         endcase
       end
                                                                                                                  clip_ve_bz_select = SRC_SELECT_FORCE_0;
                                                                                                                  clip_ve_bw_select = SRC_SELECT_FORCE_0;
10
                                                                                                           10
11
                                                                                                          11
                                                                                                                  clip_ve_a_is_wwww = 1'b0;
12 //CLIP DECODE
                                                                                                           12
                                                                                                                  clip_ve_broadcast_x = 1'b0;
13 always @(clip_plane_indx_r0 or
                                                                                                                  clip_ve_abs_a = 1'b0;
14
         clip_dst_vertex_indx_r0 or
                                                                                                                  clip_ve_abs_b = 1'b0;
15
         clip_src_vertex_indx_r0 or
                                                                                                                  clip_ve_abs_c = 1'b0;
16
         clip_src_vertex_type_r0 or
                                                                                                                  clip_ve_ax_negate = 1'b0;
17
         clip_ve_ucp_valid_r0 or
                                                                                                                  clip_ve_ay_negate = 1'b0;
18
         clip_sm_state_indx_r0 or
                                                                                                                  clip ve az negate = 1'b0;
19
         clip_state_var_indx_r0 or
                                                                                                                  clip ve aw negate = 1'b0;
20
         dx clip space def)
                                                                                                                  clip ve bx negate = 1'b0;
                                                                                                           20
21
     begin
                                                                                                           21
                                                                                                                  clip ve by negate = 1'b0;
22
                                                                                                           22
                                                                                                                  clip_ve_bz_negate = 1'b0;
23
       clip_vte_opcode = VTE_NO_OP;
                                                                                                           23
                                                                                                                  clip_ve_bw_negate = 1'b0;
24
                                                                                                                  clip_ve_cx_negate = 1'b0;
       clip vte st indx = 2'b0;
                                                                                                           24
25
       clip ve opcode = VECTOR NO OP;
                                                                                                                  clip_ve_cy_negate = 1'b0;
                                       Page 47 of 123
                                                                                                                                                 Page 48 of 123
                                                                                                                                                                              Ex. 2115 - pa_ag.v
                                                                    Ex. 2115 - pa_ag.v
```

```
clip_ve_cz_negate = 1'b0;
2
        clip_ve_cw_negate = 1'b0;
                                                                                                                     2
                                                                                                                            if(clip\_src\_vertex\_type\_r0 == 1"b1) \ begin
        clip\_ve\_bcc\_flat\_tst = 1'b0;
                                                                                                                              get\_baryc\_ax\_select = SRC\_SELECT\_X;
        clip_ve_out_mem_sel = 3'b0;
                                                                                                                              get\_baryc\_ay\_select = SRC\_SELECT\_Y;
        clip\_ve\_out\_addr \quad = 5"b000000;
                                                                                                                              get_baryc_az_select = SRC_SELECT_Z;
        clip_ve_out_we = 4'b0000;
                                                                                                                              get\_baryc\_a\_memsel = VEA\_MEMSEL\_VEOC\_VE;
        clip_ve_accum_sel = 1'b0;
                                                                                                                            end else if(clip_src_vertex_type_r0==1'b0) begin
        clip_ve_pre_acc_we = 4'b0000;
                                                                                                                              case(clip_src_vertex_indx_r0[1:0])
10
        clip_agve_dly_vertex_store_indx = 6'b0000000;
                                                                                                                                get_baryc_ax_select = SRC_SELECT_FORCE_1;
11
        clip_agve_dly_valid_bit_set = 2'b0;
                                                                                                                                get_baryc_ay_select = SRC_SELECT_FORCE_0;
12
        clip_agve_dly_user_clip_indx = 4'b0;
                                                                                                                                get_baryc_az_select = SRC_SELECT_FORCE_0;
13
       clip_agve_dly_vv_cc_test = 1'b0;
                                                                                                                    13
14
                                                                                                                               2'b01: begin
        clip agve dly ucp cc test = 1'b0;
15
        clip\_agve\_dly\_bcc\_cc\_test \qquad = 1'b0;
                                                                                                                                get barvc ax select = SRC SELECT FORCE 0;
                                                                                                                    15
                                                                                                                                get_baryc_ay_select = SRC_SELECT_FORCE_1;
16
        clip_agve_dly_ps_ucp_cc_test = 1'b0;
                                                                                                                    16
17
        clip\_agve\_dly\_ps\_engh\_test = 1'b0;
                                                                                                                    17
                                                                                                                                get_baryc_az_select = SRC_SELECT_FORCE_0;
18
                                                                                                                    18
19
       // GetBarycCoord subroutine
                                                                                                                    19
                                                                                                                               2'h10: hegin
        get_baryc_a_memsel = VEA_MEMSEL_ZERO_FLT;
20
                                                                                                                    20
                                                                                                                                get_baryc_ax_select = SRC_SELECT_FORCE_0;
21
        get\_baryc\_ax\_select = SRC\_SELECT\_FORCE\_0;
                                                                                                                    21
                                                                                                                                get_baryc_ay_select = SRC_SELECT_FORCE_0;
22
        get_baryc_ay_select = SRC_SELECT_FORCE_0;
                                                                                                                    22
                                                                                                                                get_baryc_az_select = SRC_SELECT_FORCE_1;
23
        get_baryc_az_select = SRC_SELECT_FORCE_0;
                                                                                                                    23
24
        get_baryc_ax_negate = 1'b0;
                                                                                                                    24
                                                                                                                               2'b11: begin
25
                                                                                                                                 get_baryc_ax_select = SRC_SELECT_FORCE_1;
        get_baryc_cx_negate = 1'b0;
                                          Page 49 of 123
                                                                                                                                                              Page 50 of 123
                                                                         Ex. 2115 - pa_ag.v
                                                                                                                                                                                              Ex. 2115 - pa_ag.v
            get_baryc_ax_negate = 1'b1;
                                                                                                                            if((clip\_src\_vertex\_type\_r0 == 1'b1) \parallel
 2
            get_baryc_cx_negate = 1'b1;
                                                                                                                             (clip_src_vertex_indx_r0[6]==1'b1)) begin
            get_baryc_ay_select = SRC_SELECT_FORCE_1;
                                                                                                                              get\_pos\_a\_memsel = VEA\_MEMSEL\_VEOC\_VE;
            get\_baryc\_az\_select = SRC\_SELECT\_FORCE\_1;
                                                                                                                            end else if((clip_src_vertex_type_r0==1'b0) &&
           end
                                                                                                                                  (clip_src_vertex_indx_r0[6]==1'b0)) begin
          default: begin
                                                                                                                              get\_pos\_a\_memsel = VEA\_MEMSEL\_POS\_BUF\_VE;
           get_baryc_a_memsel = 2'bxx;
                                                                                                                            end else begin
                                                                                                                              get_pos_a_memsel = 2'bxx;
            get_baryc_ax_select = 3'bxxx;
           get_baryc_ay_select = 3'bxxx;
10
           get_baryc_az_select = 3'bxxx;
                                                                                                                    10
                                                                                                                            // end GetPosition subroutine
11
           get_baryc_ax_negate = 1'bx;
                                                                                                                    11
12
            get_baryc_cx_negate = 1'bx;
                                                                                                                    12
                                                                                                                            // GetClipDist subroutine
13
                                                                                                                            get_clipdist_a_memsel = VEA_MEMSEL_VEOC_VE;
14
                                                                                                                            if(clip_src_vertex_type_r0==1'b1) begin
                                                                                                                              get_clipdist_a_select = SRC_SELECT_W;
         get_baryc_a_memsel = 2'bxx;
                                                                                                                            end else if(clip_src_vertex_type_r0==1'b0) begin
17
                                                                                                                    17
         get_baryc_ax_select = 3'bxxx;
                                                                                                                             case(clip_src_vertex_indx_r0[1:0])
18
                                                                                                                              2'b00 : begin
         get baryc ay select = 3'bxxx;
19
                                                                                                                                get_clipdist_a_select = SRC_SELECT_X;
         get baryc az select = 3'bxxx;
20
         get baryc ax negate = 1'bx;
                                                                                                                    20
21
         get baryc cx negate = 1'bx;
                                                                                                                    21
                                                                                                                               2'b01 : begin
22
                                                                                                                    22
                                                                                                                                get\_clipdist\_a\_select = SRC\_SELECT\_Y;
23
                                                                                                                    23
        // end GetBarvcCoord subroutine
                                                                                                                               end
24
                                                                                                                    24
                                                                                                                               2'b10 : begin
25
       // GetPosition subroutine
                                                                                                                                get_clipdist_a_select = SRC_SELECT_Z;
                                          Page 51 of 123
                                                                                                                                                              Page 52 of 123
                                                                         Ex. 2115 - pa_ag.v
                                                                                                                                                                                              Ex. 2115 - pa_ag.v
```

```
end
                                                                                                                         clip_ve_az_select = get_baryc_az_select;
2
          default : begin
                                                                                                                         clip_ve_ax_negate = get_baryc_ax_negate;
            get_clipdist_a_select = 3'bxxx;
                                                                                                                         clip_ve_cx_negate = get_baryc_cx_negate;
          end
                                                                                                                         // end GetBarycCoord
         endcase
                                                                                                                         clip_agve_dly_vertex_store_indx = clip_dst_vertex_indx_r0;
       end else begin
         get_clipdist_a_select = 3'bxxx;
                                                                                                                        SMC_OUTPUT_FIRST_CLIP_POS_0,
       // end GetPosition subroutine
                                                                                                                        SMC_OUTPUT_FIRST_CLIP_POS_1,
10
                                                                                                                        SMC_OUTPUT_FIRST_CLIP_POS_2,
11
                                                                                                                        SMC\_OUTPUT\_REST\_CLIP\_POS:begin
       case (clip_sm_state_indx_r0)
12
                                                                                                               12
                                                                                                                         clip_vte_opcode = VTE_CLIP_POS;
13
         SMC_OUTPUT_FIRST_BARYC_0,
                                                                                                                         clip_ve_ax_select = SRC_SELECT_X;
14
         SMC_OUTPUT_FIRST_BARYC_1,
                                                                                                                         clip_ve_ay_select = SRC_SELECT_Y;
15
         SMC_OUTPUT_FIRST_BARYC_2,
                                                                                                              15
                                                                                                                         clip ve az select = SRC SELECT Z;
         SMC_OUTPUT_REST_BARYC : begin
16
                                                                                                                         clip ve aw select = SRC SELECT W;
                                                                                                               16
17
                                                                                                              17
          if (clip ve ucp valid r0) begin
                                                                                                                         clip ve a memsel = get pos a memsel;
18
           clip_vte_opcode = VTE BC NO W;
                                                                                                              18
                                                                                                                         clip agve dly vertex store indx = clip dst vertex indx r0;
19
          end else begin
                                                                                                              19
           clip_vte_opcode = VTE BC MULT W:
20
                                                                                                              20
                                                                                                                       SMC T BLEND PREV ABC 0: begin
21
          end
                                                                                                              21
22
          // GetBarvcCoord
                                                                                                              22
                                                                                                                         clip ve opcode = VE MULTIPLY:
23
          clip_ve_a_memsel = get_baryc_a_memsel;
                                                                                                              23
                                                                                                                         clip_ve_pre_acc_we = 4'b0111;
24
          clip_ve_ax_select = get_baryc_ax_select;
                                                                                                              24
                                                                                                                         // GetBarycCoord
          clip_ve_ay_select = get_baryc_ay_select;
                                                                                                                         clip_ve_a_memsel = get_baryc_a_memsel;
                                        Page 53 of 123
                                                                                                                                                       Page 54 of 123
                                                                      Ex. 2115 - pa_ag.v
                                                                                                                                                                                     Ex. 2115 - pa_ag.v
                                                                                                                         // end GetBarycCoord
          clip ve ax select = get baryc ax select;
 2
                                                                                                                         clip ve bx select = SRC SELECT Y;
          clip_ve_ay_select = get_baryc_ay_select;
                                                                                                                         clip_ve_by_select = SRC_SELECT_Y;
          clip_ve_az_select = get_baryc_az_select;
                                                                                                                         clip ve bz select = SRC SELECT Y;
          clip_ve_ax_negate = get_baryc_ax_negate;
          clip\_ve\_cx\_negate = get\_baryc\_cx\_negate;
                                                                                                                         clip ve bw select = SRC SELECT FORCE 0;
                                                                                                                         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
          // end GetBarycCoord
          clip\_ve\_bx\_select = SRC\_SELECT\_X;
                                                                                                                         clip_ve_out_addr = clip_dst_vertex_indx_r0;
                                                                                                                         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
          clip\_ve\_by\_select = SRC\_SELECT\_X;
          clip_ve_bz_select = SRC_SELECT_X;
10
          clip_ve_bw_select = SRC_SELECT_FORCE_0;
                                                                                                              10
11
          clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                              11
                                                                                                                        SMC_T_BLEND_PREV_ABC_1: begin
12
                                                                                                              12
                                                                                                                         clip_ve_opcode = VE_MULTIPLY;
13
                                                                                                              13
                                                                                                                         clip_ve_pre_acc_we = 4'b0111;
14
         SMC_T_BLEND_CURR_ABC_0: begin
                                                                                                              14
                                                                                                                         // GetBarycCoord
15
          clip_ve_opcode = VE_MULTIPLY_ADD;
                                                                                                                         clip_ve_a_memsel = get_baryc_a_memsel;
16
          clip_ve_out_we = 4'b0111;
                                                                                                                         clip_ve_ax_select = get_baryc_ax_select;
17
          clip_ve_accum_sel = ACCUM_SEL_PRE_ACCUM;
                                                                                                                         clip_ve_ay_select = get_baryc_ay_select;
18
          clip ve bcc flat tst = 1'b1;
                                                                                                                         clip ve az select = get baryc az select;
19
          // GetBarycCoord
                                                                                                                         clip ve ax negate = get baryc ax negate;
20
          clip ve a memsel = get baryc a memsel;
                                                                                                                         clip ve cx negate = get barvc cx negate;
21
                                                                                                                         // end GetBarycCoord
          clip ve ax select = get baryc ax select;
22
                                                                                                              22
                                                                                                                         clip ve bx select = SRC SELECT Z;
          clip ve ay select = get baryc ay select;
23
                                                                                                              23
                                                                                                                         clip ve by select = SRC SELECT Z;
          clip ve az select = get baryc az select;
24
                                                                                                                         clip_ve_bz_select = SRC_SELECT_Z:
          clip_ve_ax_negate = get_baryc_ax_negate;
                                                                                                              24
          clip_ve_cx_negate = get_baryc_cx_negate;
                                                                                                                         clip ve bw select = SRC SELECT FORCE 0;
                                        Page 55 of 123
                                                                                                                                                       Page 56 of 123
                                                                      Ex. 2115 - pa_ag.v
                                                                                                                                                                                     Ex. 2115 - pa_ag.v
```

```
clip_ve_b_memsel = VEB MEMSEL CLIPTEMP VEC:
                                                                                                                SMC T BLEND PREV POS 0: begin
 1
2
                                                                                                                 clip\_ve\_opcode \quad = VE\_MULTIPLY;
 3
                                                                                                                 clip_ve_pre_acc_we = 4'b1111;
 4
        SMC_T_BLEND_CURR_ABC_1: begin
                                                                                                                 clip_ve_ax_select = SRC_SELECT_X;
         clip_ve_opcode = VE_MULTIPLY_ADD;
                                                                                                                 clip_ve_ay_select = SRC_SELECT_Y;
         clip_ve_out_we = 4'b0111;
                                                                                                                 clip_ve_az_select = SRC_SELECT_Z;
                                                                                                                 clip_ve_aw_select = SRC_SELECT_W;
         clip_ve_accum_sel = ACCUM_SEL_PRE_ACCUM;
         clip_ve_bcc_flat_tst = 1'b1;
         // GetBarycCoord
                                                                                                                 clip_ve_a_memsel = get_pos_a_memsel;
10
         clip_ve_a_memsel = get_baryc_a_memsel;
11
         clip_ve_ax_select = get_baryc_ax_select;
                                                                                                                 clip_ve_bx_select = SRC_SELECT_X;
12
         clip_ve_ay_select = get_baryc_ay_select;
                                                                                                                 clip_ve_by_select = SRC_SELECT_X;
13
                                                                                                                 clip_ve_bz_select = SRC_SELECT_X;
         clip ve az select = get baryc az select;
14
                                                                                                                 clip_ve_bw_select = SRC_SELECT_X;
         clip ve ax negate = get baryc ax negate;
                                                                                                                 clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
15
         clip ve cx negate = get baryc cx negate;
                                                                                                        15
16
         // end GetBarycCoord
                                                                                                        16
17
         clip ve bx select = SRC SELECT W:
                                                                                                        17
18
         clip ve by select = SRC SELECT W:
                                                                                                        18
                                                                                                                SMC T BLEND CURR POS 0: begin
                                                                                                                 clip_ve_opcode = VE_MULTIPLY_ADD;
19
         clip ve bz select = SRC SELECT W;
                                                                                                        19
         clip_ve_bw_select = SRC_SELECT_FORCE 0;
                                                                                                                 clip_ve_out_we = 4'b1111;
20
                                                                                                        20
         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
21
                                                                                                        21
                                                                                                                 clip ve accum sel = ACCUM SEL PRE ACCUM;
22
         clip_ve_out_addr = clip_dst_vertex_indx_r0;
                                                                                                        22
                                                                                                                 clip ve bcc flat tst = 1'b1;
23
         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                        23
                                                                                                                 clip_ve_ax_select = SRC_SELECT_X;
24
                                                                                                        24
                                                                                                                 clip_ve_ay_select = SRC_SELECT_Y;
25
                                                                                                        25
                                                                                                                 clip_ve_az_select = SRC_SELECT_Z;
                                      Page 57 of 123
                                                                                                                                              Page 58 of 123
                                                                  Ex. 2115 - pa_ag.v
                                                                                                                                                                          Ex. 2115 - pa_ag.v
          clip_ve_aw_select = SRC_SELECT_W;
                                                                                                                 clip_ve_by_select = SRC_SELECT_Z;
 2
         // GetPosition
                                                                                                                 clip ve bz select = SRC SELECT Z;
                                                                                                                 clip_ve_bw_select = SRC_SELECT_Z;
          clip ve a memsel = get pos a memsel;
                                                                                                                 clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
         // end GetPosition
         clip_ve_bx_select = SRC_SELECT_Y;
                                                                                                         5
         clip_ve_by_select = SRC_SELECT_Y;
         clip_ve_bz_select = SRC_SELECT_Y;
                                                                                                                SMC T BLEND CURR POS 1: begin
         clip ve bw select = SRC SELECT Y;
                                                                                                                 clip_ve_opcode = VE_MULTIPLY_ADD;
         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
                                                                                                                 clip_ve_out_we = 4'b1111;
10
         clip_ve_out_addr = { 2'b01,
                                                                                                        10
                                                                                                                 clip_ve_accum_sel = ACCUM_SEL_PRE_ACCUM;
11
                    clip_dst_vertex_indx_r0[3:0] };
                                                                                                        11
                                                                                                                 clip_ve_bcc_flat_tst = 1'b1;
12
         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                        12
                                                                                                                 clip_ve_ax_select = SRC_SELECT_X;
13
                                                                                                                 clip_ve_ay_select = SRC_SELECT_Y;
14
                                                                                                                 clip_ve_az_select = SRC_SELECT_Z;
15
        SMC_T_BLEND_PREV_POS_1: begin
                                                                                                                 clip_ve_aw_select = SRC_SELECT_W;
         clip_ve_opcode = VE_MULTIPLY;
16
17
         clip_ve_pre_acc_we = 4'b1111;
                                                                                                                 clip_ve_a_memsel = get_pos_a_memsel;
18
         clip_ve_ax_select = SRC_SELECT_X;
                                                                                                                 // end GetPosition
19
         clip_ve_ay_select = SRC_SELECT_Y;
                                                                                                                 clip_ve_bx_select = SRC_SELECT_W;
         clip_ve_az_select = SRC_SELECT_Z;
                                                                                                                 clip_ve_by_select = SRC_SELECT_W;
20
                                                                                                        20
                                                                                                                 clip_ve_bz_select = SRC_SELECT_W;
21
         clip ve aw select = SRC SELECT W;
                                                                                                        21
22
                                                                                                        22
                                                                                                                 clip_ve_bw_select = SRC_SELECT_W;
         // GetPosition
23
         clip_ve_a_memsel = get_pos_a_memsel;
                                                                                                        23
                                                                                                                 clip_ve_bw_select = SRC_SELECT_W;
24
                                                                                                                 clip ve out mem sel = VE OUT VEOC VECTOR BACK;
         // end GetPosition
                                                                                                        24
         clip ve bx select = SRC SELECT Z;
                                                                                                                 clip ve out addr = { 2'b01,
                                      Page 59 of 123
                                                                                                                                             Page 60 of 123
                                                                  Ex. 2115 - pa_ag.v
                                                                                                                                                                          Ex. 2115 - pa_ag.v
```

```
clip_dst_vertex_indx_r0[3:0] };
                                                                                                        1
                                                                                                                 clip_ve_bx_select = SRC_SELECT_FORCE_1;
2
         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                        2
3
                                                                                                        3
4
                                                                                                        4
                                                                                                                SMC_CLIP_T_FACTOR_PREV_0: begin
        SMC EDGE DISTANCE 0: begin
                                                                                                                clip_ve_opcode = VE_MULTIPLY;
         clip_ve_opcode = VE_MULTIPLY;
                                                                                                                clip_ve_out_mem_sel = VE_OUT_CLIPTEMP_VECTOR;
                                                                                                                clip_ve_out_we = 4'b0001;
         clip_ve_pre_acc_we = 4'b0001;
         clip_ve_abs_a = 1'b1;
                                                                                                                clip_ve_abs_a = 1'b1;
         clip_ve_ax_select = get_clipdist_a_select;
                                                                                                                clip_ve_ax_select = get_clipdist_a_select;
11
         clip_ve_a_memsel = get_clipdist_a_memsel;
                                                                                                                clip_ve_a_memsel = get_clipdist_a_memsel;
12
         // end GetClipDist
                                                                                                                // end GetClipDist
13
        clip_ve_bx_select = SRC_SELECT_FORCE_1;
                                                                                                                clip_ve_bx_select = SRC_SELECT_X;
14
                                                                                                                clip_ve_b_memsel = VEB_MEMSEL_INV_RET_SC;
15
                                                                                                       15
        SMC_EDGE_DISTANCE_1: begin
16
                                                                                                       16
17
         clip ve opcode = VE MULTIPLY ADD;
                                                                                                       17
                                                                                                               SMC_CLIP_T_FACTOR_CURR_0: begin
                                                                                                                clip ve opcode = VE MULTIPLY;
18
         clip ve out mem sel = VE OUT INVERSE:
                                                                                                       18
         clip_ve_out_we = 4'b0001;
19
                                                                                                       19
                                                                                                                clip ve out mem sel = VE OUT CLIPTEMP VECTOR;
                                                                                                                clip_ve_out_we = 4'b0010;
         clip ve accum sel = ACCUM SEL PRE ACCUM;
20
                                                                                                       20
         clip_ve_abs_a = 1'b1;
21
                                                                                                       21
                                                                                                                clip ve abs a = 1'b1;
22
         // GetClipDist
                                                                                                       22
                                                                                                                // GetClipDist
23
         clip_ve_ax_select = get_clipdist_a_select;
                                                                                                       23
                                                                                                                 clip_ve_ay_select = get_clipdist_a_select;
24
         clip_ve_a_memsel = get_clipdist_a_memsel;
                                                                                                       24
                                                                                                                 clip_ve_a_memsel = get_clipdist_a_memsel;
         // end GetClipDist
                                                                                                       25
                                                                                                                 // end GetClipDist
                                     Page 61 of 123
                                                                                                                                             Page 62 of 123
                                                                 Ex. 2115 - pa_ag.v
                                                                                                                                                                         Ex. 2115 - pa_ag.v
         clip_ve_by_select = SRC_SELECT_X;
                                                                                                                // end GetClipDist
2
         clip ve b memsel = VEB MEMSEL INV RET SC;
                                                                                                        2
                                                                                                                 clip ve bw select = SRC SELECT Y;
                                                                                                                 clip_ve_b_memsel = VEB_MEMSEL_INV_RET_SC;
3
                                                                                                        3
4
                                                                                                        4
        SMC CLIP T FACTOR PREV 1: begin
                                                                                                               SMC CLIP DIST VV: begin
         clip_ve_opcode = VE_MULTIPLY;
                                                                                                        6
         clip\_ve\_out\_mem\_sel = VE\_OUT\_CLIPTEMP\_VECTOR;
                                                                                                                clip_ve_opcode = VE_MULTIPLY_ADD;
         clip_ve_out_we = 4'b0100;
                                                                                                                clip_ve_a_is_wwww = 1'b1;
         clip_ve_abs_a = 1'b1;
                                                                                                                 clip_ve_broadcast_x = 1'b1;
10
         // GetClipDist
                                                                                                       10
                                                                                                                 clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
11
         clip_ve_az_select = get_clipdist_a_select;
                                                                                                       11
                                                                                                                if (clip_src_vertex_type_r0) begin
12
         clip_ve_a_memsel = get_clipdist_a_memsel;
                                                                                                       12
13
         // end GetClipDist
                                                                                                       13
                                                                                                                  clip_agve_dly_bcc_cc_test = 1'b1;
         clip_ve_bz_select = SRC_SELECT_Y;
14
                                                                                                       14
                                                                                                                  clip_agve_dly_vertex_store_indx = clip_dst_vertex_indx_r0;
15
         clip_ve_b_memsel = VEB_MEMSEL_INV_RET_SC;
                                                                                                       15
                                                                                                                  clip_ve_out_addr = { 2'b00,
16
                                                                                                                             clip_dst_vertex_indx_r0[3:0] };
17
                                                                                                                end else begin
18
        SMC_CLIP_T_FACTOR_CURR_1: begin
                                                                                                       18
                                                                                                                  case(clip dst vertex indx r0[1:0])
19
        clip_ve_opcode = VE_MULTIPLY;
                                                                                                                 2'b00 : begin
        clip_ve_out_mem_sel = VE_OUT_CLIPTEMP_VECTOR;
20
                                                                                                       20
                                                                                                                   clip ve out we = 4'b0001;
        clip_ve_out_we = 4'b1000;
21
                                                                                                       21
                                                                                                                 end
22
        clip_ve_abs_a = 1'b1;
                                                                                                       22
                                                                                                                 2'b01 : begin
23
                                                                                                       23
        // GetClipDist
                                                                                                                   clip ve out we = 4'b0010;
24
        clip_ve_aw_select = get_clipdist_a_select;
                                                                                                       24
                                                                                                                   end
         clip_ve_a_memsel = get_clipdist_a_memsel;
                                                                                                       25
                                                                                                                   2'b10 : begin
                                     Page 63 of 123
                                                                                                                                             Page 64 of 123
                                                                 Ex. 2115 - pa ag.v
                                                                                                                                                                         Ex. 2115 - pa ag.v
```

```
1
              clip_ve_out_we = 4'b0100;
                                                                                                                          if(dx_clip_space_def) begin
 2
                                                                                                                           clip_ve_bx_select = SRC_SELECT_FORCE_0;
            default : begin
                                                                                                                          end else begin
             clip_ve_out_we = 4'bxxxx;
                                                                                                                           clip_ve_bx_select = SRC_SELECT_FORCE_1;
            endcase
                                                                                                                          clip_ve_ax_select = SRC_SELECT_Z;
            clip_ve_out_addr = VEOC_CLIP_DIST_ORIG;
                                                                                                                         CC_CLIP_FAR : begin
                                                                                                                         clip_ve_bx_select = SRC_SELECT_FORCE_1;
          case(clip_plane_indx_r0[2:0])
10
           CC_CLIP_FAR,
                                                                                                                         clip_ve_ax_select = SRC_SELECT_Z;
11
           CC_CLIP_RIGHT,
                                                                                                             11
12
           CC_CLIP_TOP : begin
                                                                                                             12
                                                                                                                        CC_CLIP_LEFT,
13
                                                                                                             13
                                                                                                                        CC_CLIP_RIGHT : begin
            clip ve cx negate = 1'b1;
14
                                                                                                                         clip_ve_bx_select = SRC_SELECT_X;
15
           CC_CLIP_NEAR,
                                                                                                                         clip_ve_ax_select = SRC_SELECT_X;
                                                                                                             15
           CC_CLIP_LEFT,
16
                                                                                                             16
17
           CC_CLIP_BOTTOM : begin
                                                                                                             17
                                                                                                                        CC CLIP BOTTOM,
                                                                                                                        CC CLIP TOP: begin
18
            clip_ve_cx_negate = 1'b0;
                                                                                                             18
19
                                                                                                             19
                                                                                                                         clip ve bx select = SRC SELECT Y;
           default : begin
                                                                                                                         clip_ve_ax_select = SRC_SELECT_Y;
20
                                                                                                             20
21
            clip_ve_cx_negate = 1'bx;
                                                                                                             21
22
                                                                                                             22
                                                                                                                        default : begin
23
                                                                                                             23
                                                                                                                         clip_ve_bx_select = 3'bxxx;
24
          case(clip_plane_indx_r0[2:0])
                                                                                                             24
                                                                                                                         clip_ve_ax_select = 3'bxxx;
25
           CC_CLIP_NEAR : begin
                                                                                                             25
                                        Page 65 of 123
                                                                                                                                                     Page 66 of 123
                                                                                                                                                                                   Ex. 2115 - pa_ag.v
                                                                     Ex. 2115 - pa_ag.v
                                                                                                                          2'b10 : begin
          endcase
 2
          clip_ve_aw_select = SRC_SELECT_W;
                                                                                                              2
                                                                                                                           clip_ve_out_we = 4'b0100;
          // GetPosition
                                                                                                                          end
          clip_ve_a_memsel = get_pos_a_memsel;
                                                                                                                          default : begin
          // end GetPosition
                                                                                                                           clip_ve_out_we = 4'bxxxx;
          clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                                                                                                                          end
 7
                                                                                                                         endcase
                                                                                                                         clip_ve_out_addr = VEOC_CLIP_DIST_ORIG;
 8
                                                                                                              8
         SMC_CLIP_DIST_UCP: begin
          clip_ve_opcode = VE_DOT_PRODUCT;
                                                                                                                       clip_ve_ax_select = SRC_SELECT_X;
10
                                                                                                             10
11
          clip\_ve\_out\_mem\_sel = VE\_OUT\_VEOC\_VECTOR\_BACK;
                                                                                                             11
                                                                                                                        clip\_ve\_ay\_select = SRC\_SELECT\_Y;
12
          if (clip_src_vertex_type_r0) begin
                                                                                                             12
                                                                                                                       clip_ve_az_select = SRC_SELECT_Z;
13
                                                                                                                       clip_ve_aw_select = SRC_SELECT_W;
           clip_ve_out_we
14
           clip_agve_dly_bcc_cc_test = 1'b1;
                                                                                                                       clip_ve_bx_select = SRC_SELECT_X;
15
                                                                                                                       clip_ve_by_select = SRC_SELECT_Y;
           clip_agve_dly_vertex_store_indx = clip_dst_vertex_indx_r0;
16
           clip_ve_out_addr = { 2'b00,
                                                                                                                       clip_ve_bz_select = SRC_SELECT_Z;
17
                      clip_dst_vertex_indx_r0[3:0] };
                                                                                                                       clip_ve_bw_select = SRC_SELECT_W;
18
          end else begin
                                                                                                                       clip\_agve\_dly\_user\_clip\_indx = clip\_plane\_indx\_r0;
19
                                                                                                                       // GetPosition
           case(clip dst vertex indx r0[1:0])
20
            2'b00 : begin
                                                                                                             20
                                                                                                                       clip_ve_a_memsel = get_pos_a_memsel;
21
                                                                                                             21
                                                                                                                       // end GetPosition
             clip ve out we = 4'b0001;
                                                                                                                       clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
22
                                                                                                             22
23
            2'b01 : begin
                                                                                                             23
24
                                                                                                             24
             clip_ve_out_we = 4'b0010;
25
             end
                                                                                                                      SMC_PS_CULL_RADIUS_VPORT: begin
                                        Page 67 of 123
                                                                                                                                                     Page 68 of 123
                                                                     Ex. 2115 - pa_ag.v
                                                                                                                                                                                   Ex. 2115 - pa_ag.v
```

```
clip_ve_aw_select = SRC_SELECT_W;
         clip_ve_opcode = VE_MULTIPLY;
         clip_ve_out_mem_sel = VE_OUT_CLIPTEMP_VECTOR;
 2
                                                                                                         clip ve bx select = SRC SELECT X;
         clip_ve_out_we = 4'b0010;
                                                                                                         clip_ve_by_select = SRC_SELECT_Y;
         clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
                                                                                                         clip_ve_bz_select = SRC_SELECT_Z;
         clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                                                                                                         clip_ve_bw_select = SRC_SELECT_W;
         clip_ve_ay_select = SRC_SELECT_W;
                                                                                                         clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
         clip_ve_by_select = SRC_SELECT_W;
                                                                                                         clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                                                                                                         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
                                                                                                         clip_ve_out_addr = VEOC_CLIP_DIST_ORIG;
10
        SMC_PS_CULL_RADIUS_W: begin
                                                                                                         clip_ve_out_we = 4'b0001;
11
        clip_ve_opcode = VE_MULTIPLY;
12
        clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
                                                                                                 12
13
        clip_ve_out_addr = VEOC_CLIP_DIST_ORIG;
                                                                                                 13
                                                                                                         SMC_PS_XY_RADIUS_VPORT: begin
14
        clip_ve_out_we = 4'b0010;
                                                                                                         clip_ve_opcode = VE_MULTIPLY;
        clip_ve_a_memsel = VEA_MEMSEL_POINT_BUF_SC;
                                                                                                         clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
15
                                                                                                 15
        clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                         clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
16
                                                                                                 16
17
        clip_ve_ay_select = SRC_SELECT_X;
                                                                                                 17
                                                                                                         clip_ve_ax_select = SRC_SELECT_W;
        clip_ve_by_select = SRC_SELECT_Y;
18
                                                                                                 18
                                                                                                         clip_ve_ay_select = SRC_SELECT_W;
                                                                                                         clip_ve_bx_select = SRC_SELECT_X;
19
                                                                                                 19
                                                                                                         clip_ve_by_select = SRC_SELECT_Y;
20
                                                                                                 20
        SMC_PS_CLIP_DIST_UCP: begin
                                                                                                         clip_ve_out_mem_sel = VE_OUT_CLIPTEMP_VECTOR;
21
                                                                                                 21
                                                                                                         clip_ve_out_we = 4'b0011;
22
        clip_ve_opcode = VE_DOT_PRODUCT;
                                                                                                 22
23
        clip_ve_ax_select = SRC_SELECT_X;
                                                                                                 23
24
        clip_ve_ay_select = SRC_SELECT_Y;
                                                                                                 24
25
        clip_ve_az_select = SRC_SELECT_Z;
                                                                                                         SMC_PS_XY_RADIUS_W: begin
                                   Page 69 of 123
                                                                                                                                    Page 70 of 123
                                                             Ex. 2115 - pa_ag.v
                                                                                                                                                              Ex. 2115 - pa_ag.v
         clip_ve_opcode = VE_MULTIPLY;
                                                                                                  1
 2
         clip ve a memsel = VEA MEMSEL POINT BUF SC;
                                                                                                  2
         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                        SMC PS POS UR: begin
         clip_ve_ax_select = SRC_SELECT_X;
                                                                                                         clip_ve_opcode = VE_ADD;
         clip_ve_ay_select = SRC_SELECT_X;
                                                                                                         clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
         clip_ve_bx_select = SRC_SELECT_X;
                                                                                                         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
         clip_ve_by_select = SRC_SELECT_Y;
                                                                                                         clip_ve_ax_select = SRC_SELECT_X;
         clip\_ve\_out\_mem\_sel = VE\_OUT\_CLIPTEMP\_VECTOR;
                                                                                                         clip_ve_ay_select = SRC_SELECT_Y;
         clip_ve_out_we = 4'b0011;
                                                                                                         clip_ve_az_select = SRC_SELECT_Z;
                                                                                                         clip_ve_aw_select = SRC_SELECT_W;
10
                                                                                                 10
11
                                                                                                 11
                                                                                                         clip_ve_bx_select = SRC_SELECT_X;
12
        SMC_PS_POS_UL: begin
                                                                                                 12
                                                                                                         clip_ve_by_select = SRC_SELECT_Y;
                                                                                                         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
13
        clip_ve_opcode = VE_ADD;
14
        clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
                                                                                                         clip_ve_out_addr = VEOC_PS_POS_VERT_UR;
                                                                                                 14
15
        clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                 15
                                                                                                         clip_ve_out_we = 4'b1111;
        clip_ve_ax_select = SRC_SELECT_X;
17
        clip_ve_ay_select = SRC_SELECT_Y;
18
        clip_ve_az_select = SRC_SELECT_Z;
                                                                                                 18
                                                                                                        SMC_PS_POS_LR: begin
19
        clip_ve_aw_select = SRC_SELECT_W;
                                                                                                         clip_ve_opcode = VE_ADD;
        clip_ve_bx_select = SRC_SELECT_X;
                                                                                                         clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
20
                                                                                                 20
        clip_ve_by_select = SRC_SELECT_Y;
                                                                                                         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
21
                                                                                                 21
22
                                                                                                       clip_ve_ax_select = SRC_SELECT_X;
        clip_ve_bx_negate = 1'b1;
23
        clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
                                                                                                 23
                                                                                                       clip_ve_ay_select = SRC_SELECT_Y;
                                                                                                         clip_ve_az_select = SRC_SELECT_Z;
24
        clip_ve_out_addr = VEOC_PS_POS_VERT_UL;
                                                                                                 24
        clip_ve_out_we = 4'b1111;
                                                                                                         clip ve aw select = SRC SELECT W;
                                   Page 71 of 123
                                                                                                                                    Page 72 of 123
                                                             Ex. 2115 - pa ag.v
                                                                                                                                                              Ex. 2115 - pa ag.v
```

```
clip ve bx select = SRC SELECT X;
                                                                                                               SMC PS UCP DIST UL: begin
2
         clip_ve_by_select = SRC_SELECT_Y;
                                                                                                               clip_ve_opcode = VE_DOT_PRODUCT;
                                                                                                               clip_ve_a_memsel = VEA_MEMSEL_VEOC_VE;
         clip_ve_by_negate = 1'b1;
         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
                                                                                                               clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
         clip\_ve\_out\_addr \quad = VEOC\_PS\_POS\_VERT\_LR;
                                                                                                               clip_ve_ax_select = SRC_SELECT_X;
         clip_ve_out_we = 4'b1111;
                                                                                                                clip_ve_ay_select = SRC_SELECT_Y;
                                                                                                               clip_ve_az_select = SRC_SELECT_Z;
                                                                                                               clip_ve_aw_select = SRC_SELECT_W;
        SMC_PS_POS_LL: begin
                                                                                                               clip_ve_bx_select = SRC_SELECT_X;
10
                                                                                                               clip_ve_by_select = SRC_SELECT_Y;
         clip_ve_opcode = VE_ADD;
11
         clip_ve_a_memsel = VEA_MEMSEL_POS_BUF_VE;
                                                                                                               clip_ve_bz_select = SRC_SELECT_Z;
12
         clip_ve_b_memsel = VEB_MEMSEL_CLIPTEMP_VEC;
                                                                                                               clip_ve_bw_select = SRC_SELECT_W;
13
         clip_ve_ax_select = SRC_SELECT_X;
                                                                                                               clip agve dly ps ucp cc test = l'b1;
14
         clip_ve_ay_select = SRC_SELECT_Y;
                                                                                                               clip_agve_dly_vertex_store_indx = { 4'b0000,
         clip_ve_az_select = SRC_SELECT_Z;
15
                                                                                                      15
                                                                                                                                 PS VERT UL };
         clip_ve_aw_select = SRC_SELECT_W;
16
                                                                                                      16
                                                                                                               clip agve dly user clip indx = clip plane indx r0;
17
         clip ve bx select = SRC SELECT X;
                                                                                                      17
         clip_ve_by_select = SRC_SELECT_Y;
18
                                                                                                      18
19
         clip_ve_bx_negate = 1'b1;
                                                                                                      19
                                                                                                              SMC PS UCP DIST UR begin
                                                                                                               clip_ve_opcode = VE_DOT_PRODUCT;
         clip_ve_by_negate = 1'b1;
20
                                                                                                      20
                                                                                                               clip_ve_a_memsel = VEA_MEMSEL_VEOC_VE;
         clip_ve_out_mem_sel = VE_OUT_VEOC_VECTOR_BACK;
21
                                                                                                      21
         clip_ve_out_addr = VEOC_PS_POS_VERT_LL;
                                                                                                               clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
22
                                                                                                      22
                                                                                                               clip_ve_ax_select = SRC_SELECT_X;
         clip_ve_out_we = 4'b1111;
23
                                                                                                      23
24
                                                                                                      24
                                                                                                               clip_ve_ay_select = SRC_SELECT_Y;
25
                                                                                                      25
                                                                                                                clip_ve_az_select = SRC_SELECT_Z;
                                     Page 73 of 123
                                                                                                                                            Page 74 of 123
                                                                 Ex. 2115 - pa_ag.v
                                                                                                                                                                       Ex. 2115 - pa_ag.v
         clip_ve_aw_select = SRC_SELECT_W;
                                                                                                                                 PS_VERT_LR };
                                                                                                                clip\_agve\_dly\_user\_clip\_indx \quad = clip\_plane\_indx\_r0;
 2
         clip ve bx select = SRC SELECT X;
                                                                                                       2
         clip_ve_by_select = SRC_SELECT_Y;
                                                                                                       3
         clip_ve_bz_select = SRC_SELECT_Z;
                                                                                                       4
         clip ve bw select = SRC SELECT W;
                                                                                                              SMC PS UCP DIST LL: begin
                                                                                                                clip\_ve\_opcode \qquad = VE\_DOT\_PRODUCT;
         clip_agve_dly_ps_ucp_cc_test = 1'b1;
                                                                                                               clip_ve_a_memsel = VEA_MEMSEL_VEOC_VE;
         clip_agve_dly_vertex_store_indx = { 4'b0000,
                                                                                                               clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                          PS VERT UR ):
         clip_agve_dly_user_clip_indx = clip_plane_indx_r0;
                                                                                                               clip_ve_ax_select = SRC_SELECT_X;
10
                                                                                                      10
                                                                                                               clip_ve_ay_select = SRC_SELECT_Y;
11
                                                                                                      11
                                                                                                                clip_ve_az_select = SRC_SELECT_Z;
12
        SMC_PS_UCP_DIST_LR: begin
                                                                                                      12
                                                                                                                clip_ve_aw_select = SRC_SELECT_W;
13
         clip_ve_opcode = VE_DOT_PRODUCT;
                                                                                                               clip_ve_bx_select = SRC_SELECT_X;
                                                                                                               clip_ve_by_select = SRC_SELECT_Y;
14
         clip_ve_a_memsel = VEA_MEMSEL_VEOC_VE;
15
         clip_ve_b_memsel = VEB_MEMSEL_STVE_VE;
                                                                                                               clip_ve_bz_select = SRC_SELECT_Z;
         clip_ve_ax_select = SRC_SELECT_X;
                                                                                                               clip_ve_bw_select = SRC_SELECT_W;
16
17
         clip_ve_ay_select = SRC_SELECT_Y;
                                                                                                                clip_agve_dly_ps_ucp_cc_test = 1'b1;
18
         clip_ve_az_select = SRC_SELECT_Z;
                                                                                                                clip_agve_dly_vertex_store_indx = { 4'b0000,
19
         clip_ve_aw_select = SRC_SELECT_W;
                                                                                                                                 PS_VERT_LL };
         clip_ve_bx_select = SRC_SELECT_X;
20
                                                                                                      20
                                                                                                                clip agve dly user clip indx = clip plane indx r0;
         clip_ve_by_select = SRC_SELECT_Y;
21
                                                                                                      21
                                                                                                               if (clip ve ucp valid r0) begin
                                                                                                                clip_agve_dly_valid_bit_set = VE_PS_UCP_VALID;
22
         clip_ve_bz_select = SRC_SELECT_Z;
                                                                                                      22
23
         clip_ve_bw_select = SRC_SELECT_W;
                                                                                                      23
                                                                                                               end
24
         clip_agve_dly_ps_ucp_cc_test = 1'b1;
                                                                                                      24
                                                                                                               end
         clip agve dly vertex store indx = { 4'b0000,
                                                                                                      25
                                     Page 75 of 123
                                                                                                                                            Page 76 of 123
                                                                 Ex. 2115 - pa_ag.v
                                                                                                                                                                       Ex. 2115 - pa_ag.v
```

1	SMC_PS_ENGH_TEST: begin	1	ccg_ve_ax_select_rl or		
2	clip_ve_opcode = VE_CLIP_CODE_VV_ZWW;	2	ccg_ve_ay_select_rl or		
3	clip_ve_a_is_wwww = 1'b1;	3	ccg_ve_az_select_rl or		
4	clip_agve_dly_ps_engh_test = l'b1;	4	ccg_ve_aw_select_rl or		
5	clip_agve_dly_user_clip_indx = clip_plane_indx_r0;	5	ccg_ve_bx_select_rl or		
6	clip_agve_dly_vertex_store_indx = clip_src_vertex_indx_r0;	6	ccg_ve_by_select_rl or		
7	clip_ve_az_select = SRC_SELECT_X;	7	ccg_ve_bz_select_rl or		
8	clip_ve_aw_select = SRC_SELECT_Y;	8	ccg_ve_bw_select_rl or		
9	clip_ve_a_memsel = VEA_MEMSEL_VEOC_VE;	9	ccg_ve_a_is_wwww_rl or		
10	if (clip_ve_ucp_valid_r0) begin	10	ccg_ve_broadcast_x_rl or		
11	clip_agve_dly_valid_bit_set = VE_PS_ENGH_VALID;	11	ccg_ve_abs_a_rl or		
12	end	12	ccg_ve_abs_b_rl or		
13	end	13	ccg_ve_abs_c_rl or		
14		14	ccg_ve_ax_negate_rl or		
15	default;	15	ccg_ve_ay_negate_r1 or		
16	endcase	16	ccg_ve_az_negate_rl or		
17	end	17	ccg_ve_aw_negate_rl or		
18		18	ccg_ve_bx_negate_rl or		
19	//Arbiter selection of CCG or CLIP command	19	ccg_ve_by_negate_rl or		
20	always @(20	ccg_ve_bz_negate_rl or		
21	ccg_vte_opcode_rl or	21	ccg_ve_bw_negate_rl or		
22	ccg_vte_st_indx_rl or	22	ccg_ve_cx_negate_r1 or		
23	ccg_ve_opcode_rl or	23	ccg_ve_cy_negate_r1 or		
24	ccg_ve_a_memsel_rl or	24	ccg_ve_cz_negate_rl or		
25	ccg_ve_b_memsel_r1 or	25	ccg_ve_cw_negate_rl or		
	Page 77 of 123			Page 78 of 123	
	Ex. 2115 - pa_ag.v				Ex. 2115 - pa_ag.v
1	ccg_ve_bcc_flat_tst_rl or	1	clip_ve_bx_select_rl or		
2	ccg_ve_out_mem_sel_r1 or	2	clip_ve_by_select_rl or		
2	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or	2 3	clip_ve_by_select_rl or clip_ve_bz_select_rl or		
2 3 4	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or	2 3 4	clip_ve_by_select_rl or clip_ve_bz_select_rl or clip_ve_bw_select_rl or		
2 3 4 5	ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or	2 3 4 5	clip_ve_by_select_rl or clip_ve_bz_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or		
2 3 4 5 6	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or	2 3 4 5 6	clip_ve_by_select_rl or clip_ve_bz_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or		
2 3 4 5 6 7	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or	2 3 4 5 6 7	clip_ve_by_select_rl or clip_ve_bz_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or		
2 3 4 5 6 7 8	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_valid_bit_set_rl or	2 3 4 5 6 7 8	clip_ve_by_select_rl or clip_ve_bz_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or		
2 3 4 5 6 7 8	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_valid_bit_set_rl or ccg_agve_dly_user_clip_indx_rl or	2 3 4 5 6 7 8 9	clip_ve_by_select_rl or clip_ve_bz_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_an_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_c_rl or		
2 3 4 5 6 7 8 9	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_valid_bit_set_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_vvec_test_rl or	2 3 4 5 6 7 8 9	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_c_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or		
2 3 4 5 6 7 8 9 10	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_veltex_store_indx_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_vvc_ctest_rl or ccg_agve_dly_ucc_ctest_rl or ccg_agve_dly_ucc_ctest_rl or	2 3 4 5 6 7 8 9 10	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_andeast_x_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or		
2 3 4 5 6 7 8 9 10 11	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_veltex_store_indx_rl or ccg_agve_dly_uer_clip_indx_rl or ccg_agve_dly_vv_cc_test_rl or ccg_agve_dly_uep_cc_test_rl or ccg_agve_dly_uep_cc_test_rl or ccg_agve_dly_bcc_cc_test_rl or	2 3 4 5 6 7 8 9 10	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_ans_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_ay_negate_rl or clip_ve_ay_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_ve_ctest_rl or ccg_agve_dly_user_ot_strl or ccg_agve_dly_user_ot_strl or ccg_agve_dly_bc_ot_ctest_rl or ccg_agve_dly_bc_ot_ctest_rl or ccg_agve_dly_ps_user_cctest_rl or ccg_agve_dly_ps_user_cctest_rl or	2 3 4 5 6 7 8 9 10 11 12 13	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_ay_negate_rl or clip_ve_ay_negate_rl or clip_ve_ay_negate_rl or clip_ve_ay_negate_rl or clip_ve_aw_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_veltex_store_indx_rl or ccg_agve_dly_uer_clip_indx_rl or ccg_agve_dly_vv_cc_test_rl or ccg_agve_dly_uep_cc_test_rl or ccg_agve_dly_uep_cc_test_rl or ccg_agve_dly_bcc_cc_test_rl or	2 3 4 5 6 7 8 9 10 11 12 13	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_alis_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15	ccg_ve_out_mem_sel_rl or ccg_ve_out_addr_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_valid_bit_set_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_bec_oc_test_rl or ccg_agve_dly_bec_oc_test_rl or ccg_agve_dly_ps_engh_test_rl or ccg_agve_dly_ps_engh_test_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_alsa_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_by_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_be_cc_test_rl or ccg_agve_dly_be_cc_test_rl or ccg_agve_dly_be_cc_test_rl or ccg_agve_dly_be_sucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_pre_acc_we_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_bec_cc_test_rl or ccg_agve_dly_bec_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or clip_xfc_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bw_select_rl or clip_ve_a_is_wwww_rl or clip_ve_aris_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_user_clip_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_bec_cc_test_rl or ccg_agve_dly_bec_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or clip_xfc_rl or clip_vtc_opcode_rl or clip_vtc_opcode_rl or clip_vtc_st_indx_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_eye_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vc_ctest_rl or ccg_agve_dly_bec_ctest_rl or ccg_agve_dly_bec_ctest_rl or ccg_agve_dly_bec_ctest_rl or ccg_agve_dly_ps_ucp_cc test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or clip_vte_opcode_rl or clip_ve_opcode_rl or clip_ve_opcode_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_bx_alcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or clip_ve_cy_negate_rl or clip_ve_cy_negate_rl or clip_ve_cy_negate_rl or clip_ve_cy_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vc_ctest_rl or ccg_agve_dly_be_cc_test_rl or ccg_agve_dly_be_cc_test_rl or ccg_agve_dly_be_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or clip_vte_opcode_rl or clip_vte_opcode_rl or clip_ve_amemsel_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_bs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_cx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vect_est_rl or ccg_agve_dly_vc_ct_est_rl or ccg_agve_dly_bec_ce_test_rl or ccg_agve_dly_bec_ce_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or clip_vte_opcode_rl or clip_ve_opcode_rl or clip_ve_a_memsel_rl or clip_ve_a_memsel_rl or clip_ve_b_memsel_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_ais_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_bs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_cy_negate_rl or clip_ve_cx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vertex_store_indx_rl or ccg_agve_dly_vec_test_rl or ccg_agve_dly_vc_ctest_rl or ccg_agve_dly_bc_cc_test_rl or ccg_agve_dly_bc_cc_test_rl or ccg_agve_dly_bc_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_ucp_cc_test_rl or ccg_agve_dly_ps_engh_test_rl or ccg_agve_dly_ps_engh_test_rl or clip_xfc_rl or clip_vt_opcode_rl or clip_ve_a_memsel_rl or clip_ve_a_memsel_rl or clip_ve_b_memsel_rl or clip_ve_b_memsel_rl or clip_ve_b_memsel_rl or clip_ve_b_memsel_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_ais_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_bs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_c_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_bx_negate_rl or clip_ve_cy_negate_rl or clip_ve_cx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vec_test_rl or ccg_aye_dly_vc_ctest_rl or ccg_aye_dly_bc_cc_test_rl or ccg_aye_dly_bc_cc_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_engh_test_rl or clip_xfc_rl or clip_vt_opcode_rl or clip_vt_e_st_indx_rl or clip_ve_a_memsel_rl or clip_ve_b_memsel_rl or clip_ve_ax_select_rl or clip_ve_ay_select_rl or clip_ve_ay_select_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_b_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_bx_negate_rl or clip_ve_cx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vec_test_rl or ccg_aye_dly_vec_test_rl or ccg_aye_dly_bec_ce_test_rl or ccg_aye_dly_bec_ce_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_engh_test_rl or clip_xfc_rl or clip_vte_opcode_rl or clip_vte_st_indx_rl or clip_ve_a_memsel_rl or clip_ve_b_memsel_rl or clip_ve_ax_select_rl or clip_ve_ax_select_rl or clip_ve_ax_select_rl or clip_ve_az_select_rl or clip_ve_az_select_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_b_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_bx_negate_rl or clip_ve_cx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vec_test_rl or ccg_aye_dly_vc_ctest_rl or ccg_aye_dly_bc_cc_test_rl or ccg_aye_dly_bc_cc_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_engh_test_rl or clip_xfc_rl or clip_vt_opcode_rl or clip_vt_e_st_indx_rl or clip_ve_a_memsel_rl or clip_ve_b_memsel_rl or clip_ve_ax_select_rl or clip_ve_ay_select_rl or clip_ve_ay_select_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_b_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_bx_negate_rl or clip_ve_cx_negate_rl or		
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	ccg_ve_out_mem_sel_rl or ccg_ve_out_mem_sel_rl or ccg_ve_out_we_rl or ccg_ve_accum_sel_rl or ccg_ve_accum_sel_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vertex_store_indx_rl or ccg_aye_dly_vec_test_rl or ccg_aye_dly_vec_test_rl or ccg_aye_dly_bec_ce_test_rl or ccg_aye_dly_bec_ce_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_ucp_cc_test_rl or ccg_aye_dly_ps_engh_test_rl or clip_xfc_rl or clip_vte_opcode_rl or clip_vte_st_indx_rl or clip_ve_a_memsel_rl or clip_ve_b_memsel_rl or clip_ve_ax_select_rl or clip_ve_ax_select_rl or clip_ve_ax_select_rl or clip_ve_az_select_rl or clip_ve_az_select_rl or	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	clip_ve_by_select_rl or clip_ve_bx_select_rl or clip_ve_bx_select_rl or clip_ve_a_is_wwww_rl or clip_ve_broadcast_x_rl or clip_ve_abs_a_rl or clip_ve_abs_a_rl or clip_ve_abs_b_rl or clip_ve_abs_b_rl or clip_ve_ax_negate_rl or clip_ve_ax_negate_rl or clip_ve_ay_negate_rl or clip_ve_bx_negate_rl or clip_ve_cx_negate_rl or	Page 80 of 123	Ex. 2115 - pa_ag,v

```
clip ve accum sel rl or
                                                                                                                              arbsel ve bw select
                                                                                                                                                         <= ccg ve bw select rl;
 2
          clip_ve_pre_acc_we_rl or
                                                                                                                      2
                                                                                                                              arbsel_ve_a_is_wwww
                                                                                                                                                           <= ccg_ve_a_is_wwww_rl;
          clip_agve_dly_vertex_store_indx_rl or
                                                                                                                              arbsel ve broadcast x
                                                                                                                                                         <= ccg_ve_broadcast_x_rl;
          clip_agve_dly_valid_bit_set_rl or
                                                                                                                              arbsel_ve_abs_a
                                                                                                                                                       <= ccg_ve_abs_a_rl;
          clip_agve_dly_user_clip_indx_rl or
                                                                                                                              arbsel_ve_abs_b
                                                                                                                                                       <= ccg_ve_abs_b_r1;
          clip_agve_dly_vv_cc_test_rl or
                                                                                                                              arbsel_ve_abs_c
                                                                                                                                                       <= ccg ve abs c rl;
          clip_agve_dly_ucp_cc_test_rl or
                                                                                                                              arbsel_ve_ax_negate
                                                                                                                                                         <= ccg_ve_ax_negate_rl;
          clip_agve_dly_bcc_cc_test_rl or
                                                                                                                              arbsel_ve_ay_negate
                                                                                                                                                         <= ccg_ve_ay_negate_rl;
          clip_agve_dly_ps_ucp_cc_test_rl or
                                                                                                                              arbsel_ve_az_negate
                                                                                                                                                         <= ccg_ve_az_negate_rl;
10
          clip_agve_dly_ps_engh_test_rl
                                                                                                                              arbsel_ve_aw_negate
                                                                                                                                                         <= ccg_ve_aw_negate_rl;
11
                                                                                                                     11
                                                                                                                              arbsel_ve_bx_negate
                                                                                                                                                         <= ccg_ve_bx_negate_rl;
12
      begin
                                                                                                                      12
                                                                                                                              arbsel_ve_by_negate
                                                                                                                                                         <= ccg_ve_by_negate_rl;
13
                                                                                                                      13
                                                                                                                              arbsel ve bz negate
                                                                                                                                                         <= ccg ve bz negate rl;
14
        arbsel vte opcode
                                  <= ccg vte opcode rl;
                                                                                                                              arbsel ve bw negate
                                                                                                                                                         <= ccg ve bw negate rl;
15
                                 <= ccg_vte_st_indx_rl;
                                                                                                                     15
                                                                                                                                                        <= ccg_ve_cx_negate_r1;
        arbsel vte st indx
                                                                                                                             arbsel ve cx negate
16
                                  <= ccg_ve_opcode_r1;
                                                                                                                                                         <= ccg_ve_cy_negate_rl;
        arbsel ve opcode
                                                                                                                      16
                                                                                                                             arbsel ve cy negate
17
                                                                                                                     17
        arbsel ve a memsel
                                   <= ccg ve a memsel r1;
                                                                                                                             arbsel ve cz negate
                                                                                                                                                         <= ccg ve cz negate rl;
18
        arbsel ve b memsel
                                   <= ccg ve b memsel rl;
                                                                                                                     18
                                                                                                                                                         <= ccg ve cw negate r1;
                                                                                                                             arbsel ve cw negate
19
        arbsel ve ax select
                                  <= ccg ve ax select rl;
                                                                                                                     19
                                                                                                                             arbsel ve bcc flat tst
                                                                                                                                                        <= ccg ve bcc flat tst rl;
                                                                                                                                                        <= ccg_ve_out_mem_sel_rl;
20
        arbsel ve av select
                                  <= ccg ve av select rl;
                                                                                                                     20
                                                                                                                             arbsel ve out mem sel
21
        arbsel ve az select
                                  <= ccg ve az select rl;
                                                                                                                     21
                                                                                                                             arbsel ve out addr
                                                                                                                                                        <= ccg ve out addr rl;
22
        arbsel ve aw select
                                  <= ccg ve aw select rl;
                                                                                                                     22
                                                                                                                             arbsel ve out we
                                                                                                                                                        <= ccg ve out we rl;
23
        arbsel ve bx select
                                  <= ccg_ve_bx_select_rl;
                                                                                                                     23
                                                                                                                             arbsel_ve_accum_sel
                                                                                                                                                         <= ccg_ve_accum_sel_rl;
24
        arbsel_ve_by_select
                                  <= ccg_ve_by_select_rl;
                                                                                                                     24
                                                                                                                             arbsel_ve_pre_acc_we
                                                                                                                                                          <= ccg ve pre acc we rl;
25
        arbsel_ve_bz_select
                                  <= ccg_ve_bz_select_r1;
                                                                                                                     25
                                                                                                                              arbsel\_agve\_dly\_vertex\_store\_indx <= ccg\_agve\_dly\_vertex\_store\_indx\_r1;
                                          Page 81 of 123
                                                                                                                                                                Page 82 of 123
                                                                           Ex. 2115 - pa_ag.v
                                                                                                                                                                                                Ex. 2115 - pa_ag.v
        arbsel_agve_dly_valid_bit_set <= ccg_agve_dly_valid_bit_set_rl;
                                                                                                                               arbsel ve abs c
                                                                                                                                                         <= clip ve abs c rl;
2
        arbsel agve_dly_user_clip_indx <= ccg_agve_dly_user_clip_indx_rl;
                                                                                                                      2
                                                                                                                               arbsel ve ax negate
                                                                                                                                                           <= clip ve ax negate rl;
        arbsel\_agve\_dly\_vv\_cc\_test \qquad <= ccg\_agve\_dly\_vv\_cc\_test\_rl;
                                                                                                                                                           <\!\!=\!clip\_ve\_ay\_negate\_r1;
                                                                                                                               arbsel_ve_ay_negate
4
        arbsel\_agve\_dly\_ucp\_cc\_test \qquad <= ccg\_agve\_dly\_ucp\_cc\_test\_rl;
                                                                                                                               arbsel ve az negate
                                                                                                                                                           <= clip_ve_az_negate_r1;
        arbsel_agve_dly_bcc_cc_test <= ccg_agve_dly_bcc_cc_test_rl;
                                                                                                                               arbsel_ve_aw_negate
                                                                                                                                                           <= clip_ve_aw_negate_r1;
        arbsel_agve_dly_ps_ucp_cc_test <= ccg_agve_dly_ps_ucp_cc_test_rl;
                                                                                                                               arbsel_ve_bx_negate
                                                                                                                                                           <= clip_ve_bx_negate_r1;
        arbsel\_agve\_dly\_ps\_engh\_test \quad <= ccg\_agve\_dly\_ps\_engh\_test\_rl;
                                                                                                                               arbsel_ve_by_negate
                                                                                                                                                           <= clip_ve_by_negate_rl;
        if (clip xfc rl) begin
                                                                                                                               arbsel ve bz negate
                                                                                                                                                           <\!\!=\!clip\_ve\_bz\_negate\_r1;
          arbsel_vte_opcode
                                   <= clip_vte_opcode_r1;
                                                                                                                               arbsel_ve_bw_negate
                                                                                                                                                           <= clip_ve_bw_negate_r1;
10
          arbsel_vte_st_indx
                                  <= clip_vte_st_indx_r1;
                                                                                                                      10
                                                                                                                               arbsel_ve_cx_negate
                                                                                                                                                           <\!\!=clip\_ve\_cx\_negate\_r1;
11
          arbsel_ve_opcode
                                   <= clip ve opcode r1;
                                                                                                                     11
                                                                                                                               arbsel_ve_cy_negate
                                                                                                                                                           <= clip ve cy negate r1;
12
          arbsel_ve_a_memsel
                                     <= clip_ve_a_memsel_r1;
                                                                                                                     12
                                                                                                                               arbsel_ve_cz_negate
                                                                                                                                                           <= clip ve cz negate r1;
13
          arbsel_ve_b_memsel
                                     <= clip_ve_b_memsel_rl;
                                                                                                                     13
                                                                                                                                arbsel_ve_cw_negate
                                                                                                                                                           <= clip_ve_cw_negate_rl;
14
          arbsel_ve_ax_select
                                    <= clip_ve_ax_select_rl;
                                                                                                                      14
                                                                                                                               arbsel_ve_bcc_flat_tst
                                                                                                                                                          <= clip_ve_bcc_flat_tst_rl;
15
          arbsel_ve_ay_select
                                    <= clip_ve_ay_select_rl;
                                                                                                                     15
                                                                                                                                arbsel_ve_out_mem_sel
                                                                                                                                                            <= clip_ve_out_mem_sel_r1;
16
          arbsel_ve_az_select
                                    <= clip_ve_az_select_r1;
                                                                                                                                arbsel_ve_out_addr
                                                                                                                                                          <= clip\_ve\_out\_addr\_r1;
17
          arbsel_ve_aw_select
                                    <= clip_ve_aw_select_r1;
                                                                                                                      17
                                                                                                                               arbsel_ve_out_we
                                                                                                                                                          <= clip_ve_out_we_rl;
18
          arbsel ve bx select
                                    <= clip_ve_bx_select_r1;
                                                                                                                      18
                                                                                                                               arbsel ve accum sel
                                                                                                                                                           <= clip_ve_accum_sel_r1;
19
          arbsel_ve_by_select
                                    <= clip_ve_by_select_r1;
                                                                                                                                                           <= clip_ve_pre_acc_we_rl;
                                                                                                                               arbsel ve pre acc we
20
                                    <= clip_ve_bz_select_rl;
          arbsel ve bz select
                                                                                                                     20
                                                                                                                               arbsel agve dly vertex store indx <= clip agve dly vertex store indx r1;
21
                                    <= clip_ve_bw_select_rl;
                                                                                                                               arbsel agve dly valid bit set <= clip agve dly valid bit set rl;
          arbsel ve bw select
                                                                                                                     21
22
                                                                                                                     22
                                                                                                                               arbsel_agve_dly_user_clip_indx <= clip_agve_dly_user_clip_indx_rl;
          arbsel ve a is wwww
                                      <= clip ve a is wwww rl:
23
                                    <= clip ve broadcast x rl:
                                                                                                                     23
                                                                                                                               arbsel_agve_dly_vv_cc_test <= clip_agve_dly_vv_cc_test_rl;
          arbsel ve broadcast x
24
                                                                                                                     24
                                                                                                                               arbsel agve dly ucp cc test <= clip agve dly ucp cc test r1;
          arbsel ve abs a
                                   <= clip ve abs a rl;
25
          arbsel ve abs b
                                   <= clip ve abs b rl;
                                                                                                                               arbsel\_agve\_dly\_bcc\_cc\_test \qquad <= clip\_agve\_dly\_bcc\_cc\_test\_rl;
                                          Page 83 of 123
                                                                                                                                                                Page 84 of 123
                                                                          Ex. 2115 - pa ag.v
                                                                                                                                                                                                Ex. 2115 - pa ag.v
```

```
arbsel\_agve\_dly\_ps\_ucp\_cc\_test \  \  <= clip\_agve\_dly\_ps\_ucp\_cc\_test\_r1;
                                                                                                                        default:
2
         arbsel\_agve\_dly\_ps\_engh\_test \quad \mathrel{<=} clip\_agve\_dly\_ps\_engh\_test\_rl;
                                                                                                                      endcase
3
                                                                                                                3
                                                                                                                     end
4
                                                                                                                5 always @(ve_b_memsel_r2 or
 6 always @(ve_a_memsel_r2 or
                                                                                                                         stve_rdata_r2 or
         ve_veoc_vector_back_rdata_r2 or
                                                                                                                         ve_cliptemp_vec or
                                                                                                                         inv_ret_sc_data)
          pntsz_rdata_r2)
10
                                                                                                                      bmem_sel_data = 128'h0;
11
        amem_sel_data = 128'h0;
                                                                                                               11
12
                                                                                                               12
                                                                                                                      case(ve_b_memsel_r2)
13
                                                                                                               13
                                                                                                                       VEB_MEMSEL_STVE_VE: begin
       case(ve a memsel r2)
14
         VEA_MEMSEL_VEOC_VE: begin
                                                                                                                        bmem_sel_data = stve_rdata_r2;
15
                                                                                                               15
         amem_sel_data = ve_veoc_vector_back_rdata_r2;
                                                                                                                        VEB_MEMSEL_CLIPTEMP_VEC: begin
16
                                                                                                               16
17
         VEA MEMSEL POS BUF VE: begin
                                                                                                               17
                                                                                                                        bmem_sel_data = ve_cliptemp_vec;
18
                                                                                                               18
          amem sel data = pos rdata r2;
19
                                                                                                               19
                                                                                                                        VEB_MEMSEL_INV_RET_SC: begin
         VEA MEMSEL POINT BUE SC: begin
20
                                                                                                               20
                                                                                                                         bmem sel data = {64'h0,inv ret sc data};
21
         amem_sel_data = pntsz_rdata_r2;
                                                                                                               21
22
                                                                                                               22
                                                                                                                        VEB MEMSEL ZERO FLT: begin
23
         VEA MEMSEL ZERO FLT: begin
                                                                                                               23
                                                                                                                        bmem_sel_data = 128'h0;
24
         amem_sel_data = 128'h0;
                                                                                                               24
25
                                                                                                               25
                                                                                                                        default:
                                        Page 85 of 123
                                                                                                                                                       Page 86 of 123
                                                                      Ex. 2115 - pa_ag.v
                                                                                                                                                                                     Ex. 2115 - pa_ag.v
        endcase
                                                                                                                l begin
2
                                                                                                                     clip xfc r0 <= arb clip xfc;
      end
4 //swizzle data
                                                                                                                     //register ccg values for decode purposes
 5 assign agswz_ve_in_a0 = swizzle(ve_ax_select_r2, amem_sel_data);
                                                                                                                      ccg xfc r0
                                                                                                                                   <= arb_ccg_xfc;
 6 \quad \  \  assign\ agswz\_ve\_in\_a1 = swizzle(ve\_ay\_select\_r2, amem\_sel\_data);
                                                                                                                      ccg_state_var_indx_r0 <= ccg_state_var_indx;
 7 assign agswz_ve_in_a2 = swizzle(ve_az_select_r2, amem_sel_data);
                                                                                                                      ccg\_ve\_ucp\_indx\_r0 \quad \  <= ccg\_ve\_ucp\_indx;
 8 assign agswz_ve_in_a3 = swizzle(ve_aw_select_r2, amem_sel_data);
                                                                                                                     ccg_ve_cc_valid_r0 <= ccg_ve_cc_valid;
     assign agswz_ve_in_b0 = swizzle(ve_bx_select_r2, bmem_sel_data);
                                                                                                                      ccg_vertex_store_indx_r0 <= ccg_vertex_store_indx;
     assign agswz_ve_in_b1 = swizzle(ve_by_select_r2, bmem_sel_data);
                                                                                                               10
                                                                                                                      ccg\_sm\_state\_indx\_r0 \quad <= ccg\_sm\_state\_indx;
     assign agswz_ve_in_b2 = swizzle(ve_bz_select_r2, bmem_sel_data);
                                                                                                               11
12
     assign agswz_ve_in_b3 = swizzle(ve_bw_select_r2, bmem_sel_data);
                                                                                                               12
                                                                                                                     clip_plane_indx_r0 <= clip_plane_indx;
13
                                                                                                               clip_dst_vertex_indx_r0 <= clip_dst_vertex_indx;</pre>
14 always @(ve_opcode_r2)
                                                                                                                     clip_src_vertex_indx_r0 <= clip_src_vertex_indx;
                                                                                                                    clip_src_vertex_type_r0 <= clip_src_vertex_type;
                                                                                                               16 clip_ve_ucp_valid_r0 <= clip_ve_ucp_valid;
       agve_valid_op = (ve_opcode_r2!=0);
17
                                                                                                                     clip_sm_state_indx_r0 <= clip_sm_state_indx;
18
                                                                                                                     clip_state_var_indx_r0 <= clip_state_var_indx;
19
                                                                                                                     arb state var indx r0 <= nxt arb state var indx;
20
                                                                                                               20
                                                                                                                                   <= nxt_stve_re;
21 // Synchronous Section
                                                                                                               21
                                                                                                                     stve re r0
    22
                                                                                                               22
                                                                                                                                     <= nxt_state_type;
                                                                                                                      state_type_r0
23
                                                                                                               23
24 //register
                                                                                                               24
                                                                                                                     //Implement the ve_cliptemp_vector register
25 always @(posedge sclk)
                                                                                                                      if (ve_cliptemp_vector_we[0] == 1'b1) begin
                                        Page 87 of 123
                                                                                                                                                       Page 88 of 123
                                                                                                                                                                                     Ex. 2115 - pa_ag.v
                                                                      Ex. 2115 - pa_ag.v
```

```
1
         ve cliptemp vec[31:0] <= ve wdata[31:0];
                                                                                                                       gb_write_ptr <= nxt_gb_write_ptr;
2
                                                                                                                       gb_write_after_cpy <= nxt_gb_write_after_cpy;
3
       if (ve\_cliptemp\_vector\_we[1] == 1"b1) \ begin
                                                                                                                        pntsz_write_ptr <= nxt_pntsz_write_ptr;
4
         ve_cliptemp_vec[63:32] <= ve_wdata[63:32];
                                                                                                                        pntsz_write_after_cpy <= nxt_pntsz_write_after_cpy;
5
       if (ve_cliptemp_vector_we[2] == 1'b1) begin
                                                                                                                       //AG_R1 register
                                                                                                                       clip_xfc_rl <= clip_xfc_r0;
         ve\_cliptemp\_vec[95:64] \le ve\_wdata[95:64];
                                                                                                                       clip_ve_opcode_rl <= clip_ve_opcode;
        if (ve_cliptemp_vector_we[3] == 1'b1) begin
                                                                                                                       ccg_xfc_r1 <= ccg_xfc_r0;
10
         ve\_cliptemp\_vec[127:96] \le ve\_wdata[127:96];
11
                                                                                                                      ccg_ve_opcode_rl <= ccg_ve_opcode;
12
                                                                                                                      //CCG Decode R1 Delay
13
                                                                                                                13
       //maintian state management ptr's and dirty bits
                                                                                                                       ccg vte opcode rl
                                                                                                                                                 <= ccg vte opcode;
14
       ucp0_write_ptr <= nxt_ucp0_write_ptr;
                                                                                                                                                <= ccg vte st indx;
                                                                                                                       ccg vte st indx rl
15
       ucp0 write after cpy <= nxt ucp0 write after cpy;
                                                                                                                15
                                                                                                                                                <= ccg_ve_opcode;
                                                                                                                       ccg ve opcode rl
16
       ucpl write ptr <= nxt ucpl write ptr;
                                                                                                                                                  <= ccg_ve_a_memsel;
                                                                                                                16
                                                                                                                       ccg ve a memsel rl
17
                                                                                                                17
       ucpl write after cpy <= nxt ucpl write after cpy;
                                                                                                                                                  <= ccg_ve_b_memsel;
                                                                                                                       ccg ve b memsel rl
18
       ucp2 write ptr <= nxt ucp2 write ptr;
                                                                                                                18
                                                                                                                       ccg ve ax select rl
                                                                                                                                                 <= ccg ve ax select;
19
       ucp2_write_after_cpy <= nxt_ucp2_write_after_cpy;
                                                                                                                19
                                                                                                                       ccg_ve_ay_select_rl
                                                                                                                                                 <= ccg ve av select;
20
       ucp3 write ptr <= nxt ucp3 write ptr;
                                                                                                                20
                                                                                                                       ccg ve az select rl
                                                                                                                                                 <= ccg ve az select;
21
       ucp3\_write\_after\_cpy <= nxt\_ucp3\_write\_after\_cpy;
                                                                                                                21
                                                                                                                       ccg_ve_aw_select_rl
                                                                                                                                                  <= ccg_ve_aw_select;
22
       ucp4_write_ptr <= nxt_ucp4_write_ptr;
                                                                                                                22
                                                                                                                       ccg ve bx select rl
                                                                                                                                                 <= ccg ve bx select;
23
       ucp4_write_after_cpy <= nxt_ucp4_write_after_cpy;
                                                                                                                23
                                                                                                                       ccg_ve_by_select_rl
                                                                                                                                                  <= ccg_ve_by_select;
24
       ucp5_write_ptr <= nxt_ucp5_write_ptr;
                                                                                                                24
                                                                                                                       ccg_ve_bz_select_rl
                                                                                                                                                  <= ccg_ve_bz_select;
       ucp5_write_after_cpy <= nxt_ucp5_write_after_cpy;
                                                                                                                       ccg_ve_bw_select_rl
                                                                                                                                                  <= ccg_ve_bw_select;
                                        Page 89 of 123
                                                                                                                                                         Page 90 of 123
                                                                       Ex. 2115 - pa_ag.v
                                                                                                                                                                                        Ex. 2115 - pa_ag.v
                                                                                                                       ccg_agve_dly_user_clip_indx_rl <= ccg_agve_dly_user_clip_indx;
       ccg ve a is wwww rl
                                   <= ccg ve a is wwww;
2
                                                                                                                      ccg_agve_dly_vv_cc_test_rl <= ccg_agve_dly_vv_cc_test;
       ccg ve broadcast x r1
                                 <= ccg ve broadcast x;
       ccg_ve_abs_a_rl
                                <= ccg ve abs a;
                                                                                                                       ccg_agve_dly_ucp_cc_test_rl <= ccg_agve_dly_ucp_cc_test;
       ccg_ve_abs_b_rl
                                <= ccg_ve_abs_b;
                                                                                                                       ccg_agve_dly_bcc_cc_test_rl <= ccg_agve_dly_bcc_cc_test;
       ccg_ve_abs_c_rl
                                <= ccg_ve_abs_c;
                                                                                                                       ccg\_agve\_dly\_ps\_ucp\_cc\_test\_rl \quad <= ccg\_agve\_dly\_ps\_ucp\_cc\_test;
       ccg_ve_ax_negate_rl
                                 <= ccg_ve_ax_negate;
                                                                                                                        ccg_agve_dly_ps_engh_test_rl <= ccg_agve_dly_ps_engh_test;
       ccg_ve_ay_negate_rl
                                 <= ccg_ve_ay_negate;
                                                                                                                       //CLIPPER Decode R1 Delay
       ccg_ve_az_negate_r1
                                 <= ccg_ve_az_negate;
                                                                                                                       clip_vte_opcode_r1
                                                                                                                                                 <= clip vte opcode;
                                                                                                                        clip_vte_st_indx_r1
                                                                                                                                                <= clip_vte_st_indx;
       ccg_ve_aw_negate_rl
                                  <= ccg_ve_aw_negate;
10
       ccg_ve_bx_negate_rl
                                 <= ccg_ve_bx_negate;
                                                                                                                10
                                                                                                                       clip_ve_opcode_r1
                                                                                                                                                 <= clip_ve_opcode;
11
       ccg_ve_by_negate_rl
                                 <= ccg_ve_by_negate;
                                                                                                                11
                                                                                                                       clip_ve_a_memsel_rl
                                                                                                                                                  <= clip_ve_a_memsel;
12
       ccg ve bz negate rl
                                 <= ccg ve bz negate;
                                                                                                                12
                                                                                                                       clip_ve_b_memsel_rl
                                                                                                                                                  <= clip_ve_b_memsel;
13
       ccg_ve_bw_negate_rl
                                 <= ccg_ve_bw_negate;
                                                                                                                       clip_ve_ax_select_rl
                                                                                                                                                 <= clip_ve_ax_select;
14
       ccg_ve_cx_negate_rl
                                 <= ccg_ve_cx_negate;
                                                                                                                       clip_ve_ay_select_rl
                                                                                                                                                 <= clip_ve_ay_select;
15
                                 <= ccg_ve_cy_negate;
                                                                                                                       clip_ve_az_select_r1
                                                                                                                                                 <= clip_ve_az_select;
       ccg_ve_cy_negate_rl
       ccg_ve_cz_negate_rl
                                 <= ccg_ve_cz_negate;
                                                                                                                       clip_ve_aw_select_r1
                                                                                                                                                  <= clip_ve_aw_select;
17
                                                                                                                17
                                                                                                                                                 <= clip_ve_bx_select;
       ccg_ve_cw_negate_rl
                                 <= ccg_ve_cw_negate;
                                                                                                                       clip_ve_bx_select_rl
18
                                 <= ccg ve bcc flat tst;
                                                                                                                       clip ve by select rl
                                                                                                                                                 <= clip ve by select;
       ccg ve bcc flat tst rl
19
                                <= ccg_ve_out_mem_sel;
       ccg ve out mem sel rl
                                                                                                                       clip ve bz select rl
                                                                                                                                                 <= clip ve bz select;
20
                                 <= ccg_ve_out_addr;
                                                                                                                                                  <= clip_ve_bw_select;
       ccg ve out addr rl
                                                                                                                20
                                                                                                                       clip ve bw select rl
21
                                 <= ccg_ve_out_we;
                                                                                                                                                    <= clip_ve_a_is_wwww;
                                                                                                                21
                                                                                                                       clip ve a is wwww rl
       ccg ve out we rl
22
                                                                                                                                                  <= clip_ve_broadcast_x;
                                                                                                                22
                                                                                                                       clip ve broadcast x r1
       ccg ve accum sel rl
                                  <= ccg_ve_accum_sel;
23
                                <= ccg_ve_pre_acc_we;
                                                                                                                23
                                                                                                                                                 <= clip_ve_abs_a;
       ccg ve pre acc we rl
                                                                                                                       clip ve abs a rl
24
                                                                                                                                                 <= clip_ve_abs_b;
       ccg_agve_dly_vertex_store_indx_rl <= ccg_agve_dly_vertex_store_indx;
                                                                                                                24
                                                                                                                       clip ve abs b rl
       ccg agve dly valid bit set rl <= ccg agve dly valid bit set;
                                                                                                                       clip ve abs c rl
                                                                                                                                                 <= clip_ve_abs_c;
                                                                                                                                                         Page 92 of 123
                                         Page 91 of 123
                                                                       Ex. 2115 - pa_ag.v
                                                                                                                                                                                        Ex. 2115 - pa ag.v
```

```
clip_ve_ax_negate_r1
                                  <= clip_ve_ax_negate;
                                                                                                                        clip_agve_dly_ps_engh_test_rl <= clip_agve_dly_ps_engh_test;
2
       clip_ve_ay_negate_r1
                                  <= clip_ve_ay_negate;
                                                                                                                 2
       clip_ve_az_negate_r1
                                 <= clip_ve_az_negate;
                                                                                                                  3
                                                                                                                       pos re rl
                                                                                                                                              <= pos re;
       clip_ve_aw_negate_r1
                                  <= clip_ve_aw_negate;
                                                                                                                       pntsz_re_r1
                                                                                                                                              <= pntsz_re;
       clip_ve_bx_negate_r1
                                 <= clip_ve_bx_negate;
                                                                                                                       ve_veoc_vector_back_re_rl <= ve_veoc_vector_back_re;
       clip_ve_by_negate_r1
                                  <= clip_ve_by_negate;
                                                                                                                                              <= stve_re_r0;
       clip_ve_bz_negate_r1
                                 <= clip_ve_bz_negate;
                                                                                                                       //AG_R2 register
       clip_ve_bw_negate_rl
                              <= clip_ve_bw_negate;
       clip_ve_cx_negate_rl
                                 <= clip_ve_cx_negate;
                                                                                                                       vte_opcode_r2
                                                                                                                                              <= arbsel_vte_opcode;
       clip_ve_cy_negate_r1
                                 <= clip_ve_cy_negate;
                                                                                                                       vte_st_indx_r2
                                                                                                                                             <= arbsel_vte_st_indx;
11
                                 <= clip_ve_cz_negate;
                                                                                                                                              <= arbsel_ve_opcode;
       clip_ve_cz_negate_r1
                                                                                                                       ve_opcode_r2
12
       clip_ve_cw_negate_r1
                                 <= clip_ve_cw_negate;
                                                                                                                       ve_a_memsel_r2
                                                                                                                                              <= arbsel_ve_a_memsel;
13
                                 <= clip_ve_bcc_flat_tst;
                                                                                                                                               <= arbsel_ve_b_memsel;
       clip ve bcc flat tst rl
                                                                                                                       ve b memsel r2
14
                                 <= clip_ve_out_mem_sel;
                                                                                                                       ve_ax_select_r2
                                                                                                                                              <= arbsel_ve_ax_select;
       clip ve out mem sel rl
15
                                                                                                                                              <= arbsel_ve_ay_select;
                                 <= clip ve out addr;
                                                                                                                 15
                                                                                                                       ve ay select r2
       clip ve out addr rl
                                <= clip_ve_out_we;
                                                                                                                                             <= arbsel_ve_az_select;
16
                                                                                                                 16
                                                                                                                       ve az select r2
       clip ve out we rl
17
                              <= clip_ve_accum_sel;
                                                                                                                 17
                                                                                                                                              <= arbsel ve aw select:
                                                                                                                       ve aw select r2
       clip ve accum sel rl
       clip_ve_pre_acc_we_rl <= clip_ve_pre_acc_we;
18
                                                                                                                 18
                                                                                                                       ve bx select r2
                                                                                                                                              <= arbsel ve bx select:
                                                                                                                                              <= arbsel ve by select:
19
       clip_agve_dly_vertex_store_indx_rl <= clip_agve_dly_vertex_store_indx;
                                                                                                                 19
                                                                                                                        ve by select r2
       clip agve dly valid bit set rl <= clip agve dly valid bit set;
                                                                                                                                              <= arbsel ve bz select:
20
                                                                                                                 20
                                                                                                                        ve bz select r2
21
       clip\_agve\_dly\_user\_clip\_indx\_rl \quad <= clip\_agve\_dly\_user\_clip\_indx;
                                                                                                                21
                                                                                                                       ve bw select r2
                                                                                                                                              <= arbsel ve bw select:
       clip\_agve\_dly\_vv\_cc\_test\_rl \\ \hspace*{0.5cm} <\!\!= clip\_agve\_dly\_vv\_cc\_test;
22
                                                                                                                22
                                                                                                                       ve a is wwww r2
                                                                                                                                               <= arbsel ve a is wwww;
                                                                                                                                             <= arbsel_ve_broadcast_x;
23
       clip\_agve\_dly\_ucp\_cc\_test\_rl \quad \mathrel{<\!\!=} clip\_agve\_dly\_ucp\_cc\_test;
                                                                                                                23
                                                                                                                       ve_broadcast_x_r2
24
       clip_agve_dly_bcc_cc_test_rl <= clip_agve_dly_bcc_cc_test;
                                                                                                                24
                                                                                                                        ve_abs_a_r2
                                                                                                                                             <= arbsel_ve_abs_a;
       clip_agve_dly_ps_ucp_cc_test_rl <= clip_agve_dly_ps_ucp_cc_test;
                                                                                                                        ve_abs_b_r2
                                                                                                                                             <= arbsel_ve_abs_b;
                                        Page 93 of 123
                                                                                                                                                         Page 94 of 123
                                                                       Ex. 2115 - pa_ag.v
                                                                                                                                                                                        Ex. 2115 - pa_ag.v
       ve_abs_c_r2
                            <= arbsel_ve_abs_c;
                                                                                                                        agve dly ps ucp cc test r2 <= arbsel agve dly ps ucp cc test;
2
                                                                                                                        agve dly ps engh test r2 <= arbsel agve dly ps engh test;
       ve ax negate r2
                             <= arbsel ve ax negate;
                             <= arbsel_ve_ay_negate;
       ve_ay_negate_r2
4
                             <= arbsel_ve_az_negate;
                                                                                                                  4
                                                                                                                        if (pos re rl) begin
       ve az negate r2
       ve_aw_negate_r2
                              <= arbsel_ve_aw_negate;
                                                                                                                  5
                                                                                                                          pos_rdata_r2 <= pos_rdata;
       ve bx negate r2
                             <= arbsel_ve_bx_negate;
                                                                                                                  6
       ve_by_negate_r2
                              <= arbsel_ve_by_negate;
                                                                                                                 7
                                                                                                                        if(pntsz_re_r1) begin
       ve_bz_negate_r2
                             <= arbsel_ve_bz_negate;
                                                                                                                  8
                                                                                                                          pntsz rdata r2 <= pntsz rdata;
       ve_bw_negate_r2
                              <= arbsel_ve_bw_negate;
                                                                                                                  9
                                                                                                                        if(ve_veoc_vector_back_re_rl) begin
10
       ve_cx_negate_r2
                             <= arbsel_ve_cx_negate;
                                                                                                                 10
11
       ve_cy_negate_r2
                             <= arbsel_ve_cy_negate;
                                                                                                                11
                                                                                                                         ve_veoc_vector_back_rdata_r2 <= ve_veoc_vector_back_rdata;
12
       ve_cz_negate_r2
                             <= arbsel_ve_cz_negate;
                                                                                                                 12
13
                                                                                                                        if(stve_re_r1) begin
       ve_cw_negate_r2
                              <= arbsel_ve_cw_negate;
                                                                                                                 13
14
                             <= arbsel_ve_bcc_flat_tst;
                                                                                                                 14
       ve_bcc_flat_tst_r2
                                                                                                                         stve_rdata_r2 <= stve_rdata;
15
                                                                                                                15
       ve_out_mem_sel_r2 <= arbsel_ve_out_mem_sel;
                                                                                                                 16
       ve_out_addr_r2
                             <= arbsel_ve_out_addr;
17
       ve_out_we_r2 <= arbsel_ve_out_we;
                                                                                                                 17
18
       ve accum sel r2
                              <= arbsel_ve_accum_sel;
                                                                                                                 18
                                                                                                                       //AG_R3 outup register
19
                             <= arbsel_ve_pre_acc_we;
       ve pre acc we r2
                                                                                                                       ag vte opcode
                                                                                                                                            <= vte opcode r2;
                                                                                                                                           <= vte_st_indx_r2;
20
       agve dly vertex store indx r2 <= arbsel agve dly vertex store indx;
                                                                                                                 20
                                                                                                                       ag vte st indx
       agve_dly_valid_bit_set_r2 <= arbsel_agve_dly_valid_bit_set;
                                                                                                                       ag_vte_vertex_store_indx <= agve_dly_vertex_store_indx_r2[1:0];
21
                                                                                                                21
22
       agve_dly_user_clip_indx_r2 <= arbsel_agve_dly_user_clip_indx;
23
       agve_dly_vv_cc_test_r2 <= arbsel_agve_dly_vv_cc_test;
                                                                                                                 23
                                                                                                                       ag ve opcode
                                                                                                                                            <= ve opcode r2:
24
       agve_dly_ucp_cc_test_r2 <= arbsel_agve_dly_ucp_cc_test;
                                                                                                                 24
       agve dly bcc cc test r2 <= arbsel agve dly bcc cc test;
                                                                                                                       ag ve in a0
                                                                                                                                           <= agswz ve in a0;
                                                                                                                                                         Page 96 of 123
                                         Page 95 of 123
                                                                       Ex. 2115 - pa_ag.v
                                                                                                                                                                                        Ex. 2115 - pa ag.v
```

```
ag_ve_bcc_flat_tst <= ve_bcc_flat_tst_r2;
 1
     ag ve in al
                        <= agswz ve in al;
2
     ag_ve_in_a2
                        <= agswz_ve_in_a2;
                                                                                                              ag_ve_out_mem_sel <= ve_out_mem_sel_r2;
      ag_ve_in_a3
                        <= agswz ve in a3;
                                                                                                               ag_ve_out_addr <= ve_out_addr_r2;
      ag_ve_in_b0
                        <= agswz_ve_in_b0;
                                                                                                               ag_ve_out_we
                                                                                                                                  <= ve_out_we_r2;
      ag_ve_in_b1
                        <= agswz_ve_in_b1;
                                                                                                               ag_ve_accum_sel <= ve_accum_sel_r2;
                                                                                                                ag_ve_pre_accum_we <= ve_pre_acc_we_r2;
      ag_ve_in_b2
                         <= agswz_ve_in_b2;
                        <= agswz_ve_in_b3;
       ag_ve_in_b3
                                                                                                               agve_dly_valid_op_r3 <= agve_valid_op;
     ag_ve_a_is_wwww <= ve_a_is_wwww_r2;
                                                                                                              agve_dly_vertex_store_indx_r3 <= agve_dly_vertex_store_indx_r2;
     ag_ve_broadcast_x <= ve_broadcast_x_r2;
                                                                                                              agve_dly_valid_bit_set_r3 <= agve_dly_valid_bit_set_r2;
11 ag_ve_abs_a
                        <= ve_abs_a_r2;
                                                                                                         11 agve_dly_user_clip_indx_r3 <= agve_dly_user_clip_indx_r2;
12 ag_ve_abs_b
                        <= ve_abs_b_r2;
                                                                                                         12 agve_dly_vv_cc_test_r3 <= agve_dly_vv_cc_test_r2;
13 ag_ve_abs_c
                        <= ve_abs_c_r2;
                                                                                                         13 agve_dly_ucp_cc_test_r3 <= agve_dly_ucp_cc_test_r2;</pre>
14
     ag_ve_ax_negate <= ve_ax_negate_r2;
                                                                                                              agve_dly_bcc_cc_test_r3 <= agve_dly_bcc_cc_test_r2;
                                                                                                              agve_dly_ps_ucp_cc_test_r3 <= agve_dly_ps_ucp_cc_test_r2;
15
                       <= ve ay negate r2;
                                                                                                         15
     ag ve ay negate
                         <= ve_az_negate_r2;
                                                                                                               agve_dly_ps_engh_test_r3 <= agve_dly_ps_engh_test_r2;
16
     ag ve az negate
17
                         <= ve aw negate r2:
                                                                                                         17
     ag ve aw negate
18
                         <= ve bx negate r2:
                                                                                                         18
     ag ve bx negate
19
      ag ve by negate
                         <= ve_by_negate_r2;
                                                                                                        19
                                                                                                               agve dly0 <= { agve dly valid op r3,
                          <= ve bz negate r2:
20
      ag ve bz negate
                                                                                                         20
                                                                                                                      agve dly vertex store indx r3,
21
     ag_ve_bw_negate
                         <= ve bw negate r2;
                                                                                                        21
                                                                                                                       agve dly valid bit set r3,
22
     ag_ve_cx_negate
                          <= ve cx negate r2;
                                                                                                        22
                                                                                                                      agve_dly_user_clip_indx_r3,
23
      ag_ve_cy_negate
                          <= ve_cy_negate_r2;
                                                                                                         23
                                                                                                                      agve_dly_vv_cc_test_r3,
      ag_ve_cz_negate
24
                          <= ve_cz_negate_r2;
                                                                                                         24
                                                                                                                      agve_dly_ucp_cc_test_r3,
      ag_ve_cw_negate
                           <= ve_cw_negate_r2;
                                                                                                                       agve_dly_bcc_cc_test_r3,
                                      Page 97 of 123
                                                                                                                                               Page 98 of 123
                                                                  Ex. 2115 - pa_ag.v
                                                                                                                                                                           Ex. 2115 - pa_ag.v
              agve dly ps ucp cc test r3,
2
              agve_dly_ps_engh_test_r3 };
                                                                                                          3 // Functions
                                                                                                          4
      agve_dly1 <= agve_dly0;
      agve_dly2 <= agve_dly1;
      agve_dly3 <= agve_dly2;
                                                                                                          6 // Function implementation of arb
      agve\_dly4 <= agve\_dly3;
                                                                                                          7 function [1:0] arb;
       agve\_dly5 <= agve\_dly4;
                                                                                                              input srst:
                                                                                                               input arb_clip_ve_valid;
10 //add latencey matching stages here
                                                                                                         10
                                                                                                              input arb_ccg_ve_valid;
11
                                                                                                        11
                                                                                                              reg arb_nxt_clip_xfc;
12
      { agve_dly_valid_op,
                                                                                                        12
                                                                                                              reg arb_nxt_ccg_xfc;
13
                                                                                                        13 begin
       agve_dly_vertex_store_indx,
       agve_dly_valid_bit_set,
14
                                                                                                         14
                                                                                                              if ((arb_clip_ve_valid == 1'b1) && (srst == 1'b0)) begin
15
       agve_dly_user_clip_indx,
                                                                                                               arb_nxt_ccg_xfc = 1'b0;
16
       agve_dly_vv_cc_test,
                                                                                                                arb_nxt_clip_xfc = 1'b1;
17
       agve_dly_ucp_cc_test,
                                                                                                              end else if ((arb_ccg_ve_valid == 1'b1) && (srst == 1'b0)) begin
18
       agve dly bcc cc test,
                                                                                                                arb_nxt_ccg_xfc = 1'b1;
19
                                                                                                                arb_nxt_clip_xfc = 1'b0;
       agve dly ps ucp cc test,
20
       agve_dly_ps_engh_test } <= agve_dly5;
                                                                                                         20
                                                                                                              end else begin
21
                                                                                                                arb_nxt_ccg_xfc = 1'b0;
                                                                                                        21
     end
22
                                                                                                        22
                                                                                                                arb nxt clip xfc = 1'b0;
23
                                                                                                        23
   assign ag_to_clip_point_size = pntsz_rdata_r2;
24
                                                                                                         24
                                                                                                               arb = {arb_nxt_clip_xfc, arb_nxt_ccg_xfc};
25
                                                                                                         2.5
                                      Page 99 of 123
                                                                                                                                              Page 100 of 123
                                                                  Ex. 2115 - pa_ag.v
                                                                                                                                                                           Ex. 2115 - pa_ag.v
```

```
1
     endfunction
                                                                                                                   1 begin
2
                                                                                                                         if (srst) begin
3 // Function implementation of dirty bit
                                                                                                                           nxt_write_ptr = 3'b000;
4 function [0:0] dirty;
                                                                                                                          end else if (write_after_cpy && rbiu_cpy) begin
                                                                                                                           nxt\_write\_ptr = cur\_write\_ptr + 3'b001;
       input rbiu_cpy;
       input [3:0] sel;
                                                                                                                            nxt_write_ptr = cur_write_ptr;
       reg nxt_write_after_cpy;
                                                                                                                          stwtptr = nxt_write_ptr;
                                                                                                                  10
       if (srst \parallel rbiu_cpy) begin
11
                                                                                                                  11 endfunction
         nxt_write_after_cpy = 1'b0;
12
       end else if (|sel) begin
                                                                                                                  12
13
                                                                                                                  13 function [31:0] swizzle;
         nxt_write_after_cpy = 1'b1;
14
                                                                                                                         input [2:0] ve swizzle select;
                                                                                                                         input [127:0] mem_sel_data;
15
                                                                                                                  15
       dirty = nxt_write_after_cpy;
16
                                                                                                                  16
                                                                                                                         reg [31:0] swizzled data;
17
                                                                                                                  17
     endfunction
18
                                                                                                                  18
                                                                                                                          swizzled data= 32'h0:
19 // Function implementation of state write ptr stwtptr
                                                                                                                  19
20 function [2:0] stwtptr;
                                                                                                                  20
                                                                                                                          case( ve swizzle select)
21
                                                                                                                  21
                                                                                                                          SRC SELECT X: begin
22
       input rbiu_cpy;
                                                                                                                  22
                                                                                                                            swizzled_data = mem_sel_data[31:0];
23
                                                                                                                  23
24
       input [2:0] cur_write_ptr;
                                                                                                                  24
                                                                                                                           SRC_SELECT_Y: begin
       reg [2:0] nxt_write_ptr;
                                                                                                                  25
                                                                                                                            swizzled_data = mem_sel_data[63:32];
                                         Page 101 of 123
                                                                                                                                                           Page 102 of 123
                                                                        Ex. 2115 - pa_ag.v
                                                                                                                                                                                          Ex. 2115 - pa_ag.v
                                                                                                                             .r_data(agrd_dx_clip_space_def),
2
         SRC SELECT Z: begin
                                                                                                                             .st_data(dx_clip_space_def),
 3
          swizzled_data = mem_sel_data[95:64];
                                                                                                                              .st_indx(arb_state_var_indx_r0),
 4
                                                                                                                              .w addr(rbiu wa), //context
         SRC SELECT W: begin
                                                                                                                            .r\_addr(rbiu\_wd[2:0]),
          swizzled_data = mem_sel_data[127:96];
                                                                                                                              .we(rbiu_we),
7
                                                                                                                              .re(rbiu_re),
 8
         SRC SELECT FORCE 0: begin
                                                                                                                              .sel(rbiu\_ag\_dx\_clip\_sp\_def\_sel),
          swizzled_data = 32'h0;
                                                                                                                              .cpy(rbiu_cpy),
10
                                                                                                                  10
                                                                                                                              .clk(sclk));
11
         SRC_SELECT_FORCE_1: begin
                                                                                                                  11
12
           swizzled_data = 32'h3f800000;
                                                                                                                  12 ati_1rp_state_storage #(3) ustate_storage_reg_ucp0 (
13
                                                                                                                             .w_data(ucp0_write_ptr),
14
                                                                                                                             .r_data(agrd_ucp0_write_ptr),
                                                                                                                            .st_data(ucp0_rd_off),
        swizzle = swizzled_data;
                                                                                                                            .st_indx(arb_state_var_indx_r0),
17
                                                                                                                              .w_addr(rbiu_wa), //context
18
    endfunction
                                                                                                                           .r addr(rbiu wd[2:0]),
19
                                                                                                                             .we(rbiu we).
20
21
                                                                                                                  20
                                                                                                                             .re(rbiu re),
                                                                                                                  21
                                                                                                                            .sel(|rbiu_ag_ucp0_sel),
22
     // State Storage Instantiation
     22
                                                                                                                             .cpy(rbiu cpy),
                                                                                                                  23
                                                                                                                              .clk(sclk)):
25
     ati\_lrp\_state\_storage \#(1) \qquad ustate\_storage\_reg\_dx\_clip \, (
           .w_data(rbiu_wd[19]),
                                                                                                                  25 ati_1rp_state_storage #(3) ustate_storage_reg_ucp1 (
                                        Page 103 of 123
                                                                                                                                                           Page 104 of 123
                                                                        Ex. 2115 - pa_ag.v
                                                                                                                                                                                          Ex. 2115 - pa_ag.v
```

```
.w\_data(ucp\,l\_write\_ptr),
                                                                                                                           1 \quad \  ati\_lrp\_state\_storage~\#(3)~ustate\_storage\_reg\_ucp3~(
 2
            .r\_data(agrd\_ucpl\_write\_ptr),
                                                                                                                                      .w_data(ucp3_write_ptr),
            .st\_data(ucpl\_rd\_off),
                                                                                                                                      .r_data(agrd_ucp3_write_ptr),
            .st_indx(arb_state_var_indx_r0),
                                                                                                                                      .st_data(ucp3_rd_off),
             .w_addr(rbiu_wa), //context
                                                                                                                                      .st_indx(arb_state_var_indx_r0),
          .r\_addr(rbiu\_wd[2:0]),
                                                                                                                                      .w_addr(rbiu_wa), //context
            .we(rbiu_we),
                                                                                                                                   .r_addr(rbiu_wd[2:0]),
                                                                                                                                      .we(rbiu_we),
            .re(rbiu_re),
            .sel(|rbiu_ag_ucp1_sel),
                                                                                                                                      .re(rbiu_re),
10
            .cpy(rbiu_cpy),
                                                                                                                                      .sel(|rbiu\_ag\_ucp3\_sel),
11
             .clk(sclk));
                                                                                                                         11
                                                                                                                                      .cpy(rbiu_cpy),
12
                                                                                                                         12
                                                                                                                                      .clk(sclk));
13
                                                                                                                         13
    ati 1rp state storage #(3) ustate storage reg ucp2 (
14
                                                                                                                         14 ati_lrp_state_storage #(3) ustate_storage_reg_ucp4 (
            .w data(ucp2 write ptr).
15
                                                                                                                         15
            .r data(agrd ucp2 write ptr),
                                                                                                                                      .w data(ucp4 write ptr),
16
                                                                                                                         16
            .st data(ucp2 rd off),
                                                                                                                                      .r data(agrd ucp4 write ptr),
17
                                                                                                                         17
            .st indx(arb state var indx r0),
                                                                                                                                     .st_data(ucp4_rd_off).
18
            .w addr(rbiu wa). //context
                                                                                                                         18
                                                                                                                                     .st indx(arb state var indx r0),
19
          .r addr(rbiu wd[2:0]),
                                                                                                                         19
                                                                                                                                      .w_addr(rbiu_wa), //context
20
            we(rbin we)
                                                                                                                         20
                                                                                                                                   .r addr(rbiu wd[2:0]),
21
            .re(rbiu re),
                                                                                                                         21
                                                                                                                                      .we(rbiu we).
22
            .sel(|rbiu_ag_ucp2_sel),
                                                                                                                         22
                                                                                                                                      .re(rbiu re),
23
            .cpy(rbiu_cpy),
                                                                                                                         23
                                                                                                                                      .sel(|rbiu_ag_ucp4_sel),
24
             .clk(sclk));
                                                                                                                         24
                                                                                                                                      .cpy(rbiu_cpy),
25
                                                                                                                         25
                                                                                                                                      .clk(sclk));
                                           Page 105 of 123
                                                                                                                                                                    Page 106 of 123
                                                                             Ex. 2115 - pa_ag.v
                                                                                                                                                                                                      Ex. 2115 - pa_ag.v
                                                                                                                                      .clk(sclk));
 2
                                                                                                                          2
    ati_1rp_state_storage #(3) ustate_storage_reg_ucp5 (
            .w_data(ucp5_write_ptr),
                                                                                                                              ati_lrp_state_storage #(3) ustate_storage_reg_pntsz (
 4
            .r_data(agrd_ucp5_write_ptr),
                                                                                                                                      .w\_data(pntsz\_write\_ptr),
            .st_data(ucp5_rd_off),
                                                                                                                                      .r\_data(agrd\_pntsz\_write\_ptr),
            .st_indx(arb_state_var_indx_r0),
                                                                                                                                      .st_data(pntsz_rd_off),
             .w_addr(rbiu_wa), //context
                                                                                                                                      .st\_indx(arb\_state\_var\_indx\_r0),
          .r\_addr(rbiu\_wd[2:0]),
                                                                                                                                      .w_addr(rbiu_wa), //context
             .we(rbiu_we),
                                                                                                                                    .r_addr(rbiu_wd[2:0]),
                                                                                                                                      .we(rbiu_we),
10
            .re(rbiu_re),
                                                                                                                         10
11
             .sel(|rbiu_ag_ucp5_sel),
                                                                                                                         11
                                                                                                                                      .re(rbiu_re),
12
             .cpy(rbiu_cpy),
                                                                                                                         12
                                                                                                                                      .sel(|rbiu\_ag\_pntsz\_sel),
13
                                                                                                                         13
                                                                                                                                      .cpy(rbiu_cpy),
14
                                                                                                                         14
15
                                                                                                                         15
    ati_lrp_state_storage #(3) ustate_storage_reg_gb (
            .w_data(gb_write_ptr),
17
            .r_data(agrd_gb_write_ptr),
18
            .st data(gb rd off),
19
            .st_indx(arb_state_var_indx_r0),
                                                                                                                         20
                                                                                                                              // Memory Instantiation
20
            .w addr(rbiu wa), //context
                                                                                                                               21
22
21
          .r addr(rbiu wd[2:0]),
22
            .we(rbiu we).
                                                                                                                         23
                                                                                                                               `ifdef\,USE\_BEHAVE\_MEM
23
            .re(rbiu re),
                                                                                                                         24
24
            .sel(|rbiu_ag_gb_sel),
                                                                                                                              // Instantiate the cliptemp memory as
25
            .cpy(rbiu_cpy),
                                                                                                                              // 4 35d by 32w to create 35d x 128w
                                           Page 107 of 123
                                                                                                                                                                    Page 108 of 123
                                                                             Ex. 2115 - pa_ag.v
                                                                                                                                                                                                      Ex. 2115 - pa_ag.v
```

```
1 dum_mem_p2 #(
                                                                                                                      iWCLK(sclk)
 2 \quad u0\_clptmp\_ADDR\_WIDTH \, ,
                                                                                                                      .iMER(ve_veoc_vector_back_re),
 3 \quad u0\_clptmp\_DATA\_WIDTH \, ,
                                                                                                                      . iMEW (ve\_veoc\_vector\_back\_we[1]),
     u0\_clptmp\_WORDS
                                                                                                                      .iWEN(ve_veoc_vector_back_we[1]),
     u0 clptmp DEBUG
                                                                                                                      .iRADR(ve_veoc_vector_back_raddr),
 6 )
                                                                                                                      .iWADR(ve_waddr),
     u0_clptmp_dum_mem_p2 (
                                                                                                                      .iD(ve_wdata[63:32]),
           .iRCLK(sclk),
                                                                                                                      .oQ(ve_veoc_vector_back_rdata[63:32]));
           .iWCLK(sclk),
10
                                                                                                          10 dum_mem_p2 #(
           .iMER(ve_veoc_vector_back_re),
11
           .iMEW(ve_veoc_vector_back_we[0]),
                                                                                                          11 u2_clptmp_ADDR_WIDTH
12
           .iWEN(ve_veoc_vector_back_we[0]),
                                                                                                           12 u2_clptmp_DATA_WIDTH,
13
           .iRADR(ve_veoc_vector_back_raddr),
                                                                                                          13 u2_clptmp_WORDS ,
14
           .iWADR(ve_waddr),
                                                                                                           14 u2_clptmp_DEBUG
15
                                                                                                          15 )
          .iD(ve wdata[31:0]),
16
           .oQ(ve veoc vector back rdata[31:0]));
                                                                                                           16
                                                                                                              u2 clptmp dum mem p2 (
17
                                                                                                          17
                                                                                                                      .iRCLK(sclk).
                                                                                                                      .iWCLK(sclk),
18 dum mem p2 #(
                                                                                                          18
19 ul clptmp ADDR WIDTH,
                                                                                                          19
                                                                                                                      .iMER(ve_veoc_vector_back_re),
20 ul clptmp DATA WIDTH,
                                                                                                                      .iMEW(ve_veoc_vector_back_we[2]),
                                                                                                          20
21 u1_clptmp_WORDS
                                                                                                          21
                                                                                                                     .iWEN(ve veoc vector back we[2]),
22 u1_clptmp_DEBUG
                                                                                                          22
                                                                                                                     .iRADR(ve veoc vector back raddr),
23 )
                                                                                                          23
                                                                                                                     .iWADR(ve_waddr),
24 \qquad u1\_clptmp\_dum\_mem\_p2 \ (
                                                                                                          24
                                                                                                                      .iD(ve_wdata[95:64]),
25
           .iRCLK(sclk),
                                                                                                          25
                                                                                                                      .oQ(ve_veoc_vector_back_rdata[95:64]));
                                      Page 109 of 123
                                                                                                                                                 Page 110 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                              Ex. 2115 - pa_ag.v
2 dum mem p2 #(
                                                                                                              u pntsz dum mem p2 (
 3 u3_clptmp_ADDR_WIDTH,
                                                                                                                      .iRCLK(sclk).
4 u3_clptmp_DATA_WIDTH,
                                                                                                                      .iWCLK(sclk).
                                                                                                                      .iMER(pntsz_re),
     u3_clptmp_WORDS
 6 u3 clptmp DEBUG
                                                                                                                      .iMEW(pntsz_mem_we),
 7 )
                                                                                                                      . iWEN (pntsz\_mem\_we),
 8 u3_clptmp_dum_mem_p2 (
                                                                                                                      .iRADR(pntsz raddr),
           .iRCLK(sclk),
                                                                                                                      .iWADR(pntsz_mem_waddr),
10
           .iWCLK(sclk),
                                                                                                          10
                                                                                                                      .iD(pos_pntsz_ag_mem_data[31:0]),
11
           .iMER(ve_veoc_vector_back_re),
                                                                                                          11
                                                                                                                      .oQ(pntsz\_rdata));
12
           .iMEW(ve_veoc_vector_back_we[3]),
                                                                                                          12
13
           .iWEN(ve_veoc_vector_back_we[3]),
                                                                                                          13 // Instantiate the position memory
14
           .iRADR(ve_veoc_vector_back_raddr),
                                                                                                          14 // 1 64d x 128w
15
           .iWADR(ve_waddr),
                                                                                                          15 dum_mem_p2 #(
16
           .iD(ve_wdata[127:96]),
                                                                                                           16 u_pos_ADDR_WIDTH,
17
           .oQ(ve_veoc_vector_back_rdata[127:96]));
                                                                                                          17 u_pos_DATA_WIDTH,
18
                                                                                                          18 u_pos_WORDS ,
19 // Instantiate the pntsz memory
                                                                                                          19 u_pos_DEBUG
                                                                                                          20 )
20 // 1 64d x 32w
21 dum mem p2 #(
                                                                                                          21 u pos dum mem p2 (
22 u pntsz ADDR WIDTH .
                                                                                                          22
                                                                                                                      .iRCLK(sclk).
23 u_pntsz_DATA_WIDTH,
                                                                                                          23
                                                                                                                      .iWCLK(sclk).
24 u pntsz WORDS
                                                                                                          24
                                                                                                                      .iMER(pos re),
25 u pntsz DEBUG
                                                                                                                      .iMEW(pos_mem_we),
                                      Page 111 of 123
                                                                                                                                                 Page 112 of 123
                                                                   Ex. 2115 - pa_ag.v
                                                                                                                                                                              Ex. 2115 - pa_ag.v
```

```
.iWEN(pos_mem_we),
                                                                                                              1 dum_mem_p2 #(
 2
                                                                                                             2 ul_stve_ADDR_WIDTH,
           . iRADR (pos\_raddr),
            .iWADR(pos_mem_waddr),
                                                                                                              3 ul_stve_DATA_WIDTH,
            .iD(pos_pntsz_ag_mem_data),
                                                                                                              4 ul_stve_WORDS ,
            .oQ(pos_rdata));
                                                                                                              5 ul stve DEBUG
                                                                                                             6 )
 7 // Instantiate the state for ucp,gb,pntsz memory as
                                                                                                                 u1_stve_dum_mem_p2 (
 8 \hspace{0.4cm} \ensuremath{\text{//}} 4 \, 64 d by 32w to create 64d x 128w
                                                                                                                        .iRCLK(sclk),
                                                                                                                        .iWCLK(sclk),
 9 dum_mem_p2 #(
10 u0_stve_ADDR_WIDTH,
                                                                                                                        .iMER(stve_re_r0),
11 u0_stve_DATA_WIDTH,
                                                                                                                        .iMEW(stve_we[1]),
12 u0_stve_WORDS ,
                                                                                                                       .iWEN(stve_we[1]),
13 u0_stve_DEBUG
                                                                                                                        .iRADR(stve_raddr),
14 )
                                                                                                                        .iWADR(stve_wa),
15
                                                                                                                       .iD(rbiu_wd),
    u0 stve dum mem p2 (
                                                                                                            15
16
           .iRCLK(sclk),
                                                                                                            16
                                                                                                                        .oQ(stve rdata[63:32]));
17
           .iWCLK(sclk).
                                                                                                            17
18
           .iMER(stve re r0).
                                                                                                            18 dum mem p2 #(
19
           .iMEW(stve_we[0]),
                                                                                                            19 u2 stve ADDR WIDTH ,
20
           .iWEN(stve_we[0]),
                                                                                                            20 u2 stve DATA WIDTH .
21
                                                                                                            21 u2 stve WORDS .
           .iRADR(stve_raddr).
           .iWADR(stve wa),
                                                                                                            22 u2 stve DEBUG
22
23
           .iD(rbiu wd),
                                                                                                            23 )
24
            .oQ(stve_rdata[31:0]));
                                                                                                            24 \quad u2\_stve\_dum\_mem\_p2 \ (
25
                                                                                                                        .iRCLK(sclk),
                                       Page 113 of 123
                                                                                                                                                   Page 114 of 123
                                                                     Ex. 2115 - pa_ag.v
                                                                                                                                                                                 Ex. 2115 - pa_ag.v
           .iWCLK(sclk),
 2
            .iMER(stve re r0),
                                                                                                             2 dum_mem_syn_stub #(
                                                                                                                 u0\_clptmp\_ADDR\_WIDTH\ ,
            .iMEW(stve_we[2]),
            .iWEN(stve_we[2]),
                                                                                                             4 \quad u0\_clptmp\_DATA\_WIDTH \, ,
            .iRADR(stve_raddr),
                                                                                                                u0_clptmp_WORDS
            .iWADR(stve wa),
                                                                                                              6 u0_clptmp_DEBUG
            .iD(rbiu_wd),
                                                                                                             7 )
            .oQ(stve_rdata[95:64]));
                                                                                                              8 \quad \  u0\_clptmp\_dum\_mem\_p2 \ (
                                                                                                                        .iRCLK(sclk),
10 dum_mem_p2 #(
                                                                                                            10
                                                                                                                        .iWCLK(sclk),
11 u3_stve_ADDR_WIDTH,
                                                                                                            11
                                                                                                                        .iMER(ve_veoc_vector_back_re),
12 u3_stve_DATA_WIDTH,
                                                                                                            12
                                                                                                                        . iMEW (ve\_veoc\_vector\_back\_we[0]),
13 u3_stve_WORDS ,
                                                                                                                        .iWEN(ve_veoc_vector_back_we[0]),
14 u3_stve_DEBUG
                                                                                                            14
                                                                                                                        .iRADR(ve_veoc_vector_back_raddr),
15 )
                                                                                                                       .iWADR(ve_waddr),
16 u3_stve_dum_mem_p2 (
                                                                                                                        .iD(ve_wdata[31:0]),
17
           .iRCLK(sclk),
                                                                                                                        .oQ(ve_veoc_vector_back_rdata[31:0]));
18
           .iWCLK(sclk),
                                                                                                            18
19
           .iMER(stve_re_r0),
                                                                                                            19 dum_mem_syn_stub #(
20
           .iMEW(stve_we[3]),
                                                                                                            20 ul clptmp ADDR WIDTH.
21
           .iWEN(stve_we[3]),
                                                                                                            21 ul clptmp DATA WIDTH,
22
           .iRADR(stve_raddr).
                                                                                                            22 ul_clptmp_WORDS ,
23
           .iWADR(stve wa).
                                                                                                            23 ul_clptmp_DEBUG
24
           .iD(rbiu wd),
                                                                                                            24 )
25
           .oQ(stve_rdata[127:96]));
                                                                                                            25 u1_clptmp_dum_mem_p2 (
                                       Page 115 of 123
                                                                                                                                                   Page 116 of 123
                                                                     Ex. 2115 - pa_ag.v
                                                                                                                                                                                 Ex. 2115 - pa_ag.v
```

```
iRCLK(sclk)
                                                                                                                      .oQ(ve veoc vector back rdata[95:64])):
 2
           .iWCLK(sclk).
           .iMER(ve_veoc_vector_back_re),
                                                                                                           3 dum_mem_syn_stub #(
           .iMEW(ve_veoc_vector_back_we[1]),
                                                                                                           4 u3_clptmp_ADDR_WIDTH
           .iWEN(ve_veoc_vector_back_we[1]),
                                                                                                            5 \quad u3\_clptmp\_DATA\_WIDTH \, ,
           .iRADR(ve_veoc_vector_back_raddr),
                                                                                                            6 u3_clptmp_WORDS ,
           .iWADR(ve_waddr),
                                                                                                               u3_clptmp_DEBUG
           .iD(ve_wdata[63:32]),
                                                                                                           8 )
           .oQ(ve_veoc_vector_back_rdata[63:32]));
                                                                                                               u3_clptmp_dum_mem_p2 (
10
                                                                                                                      .iRCLK(sclk),
11 dum_mem_syn_stub #(
                                                                                                                      .iWCLK(sclk),
12 u2_clptmp_ADDR_WIDTH,
                                                                                                                      .iMER(ve_veoc_vector_back_re),
13 u2_clptmp_DATA_WIDTH,
                                                                                                                      .iMEW(ve_veoc_vector_back_we[3]),
14 u2_clptmp_WORDS
                                                                                                                      .iWEN(ve_veoc_vector_back_we[3]),
15 u2_clptmp_DEBUG
                                                                                                           15
                                                                                                                     .iRADR(ve veoc vector back raddr),
16 )
                                                                                                                     .iWADR(ve_waddr),
                                                                                                           16
17
                                                                                                           17
                                                                                                                      .iD(ve_wdata[127:96]).
    u2 clptmp dum mem p2 (
18
           .iRCLK(sclk).
                                                                                                           18
                                                                                                                      .oQ(ve_veoc_vector_back_rdata[127:96]));
19
           iWCLK(sclk)
                                                                                                           19
20
           .iMER(ve_veoc_vector_back_re),
                                                                                                           20 // Instantiate the pntsz memory
21
                                                                                                          21 // 1 64d x 32w
           .iMEW(ve veoc vector back we[2]),
           .iWEN(ve_veoc_vector_back_we[2]),
22
                                                                                                           22 dum mem syn stub #(
23
          .iRADR(ve_veoc_vector_back_raddr),
                                                                                                           23 u pntsz ADDR WIDTH
24
           .iWADR(ve_waddr),
                                                                                                           24 u_pntsz_DATA_WIDTH,
25
           .iD(ve_wdata[95:64]),
                                                                                                           25 u_pntsz_WORDS ,
                                      Page 117 of 123
                                                                                                                                                 Page 118 of 123
                                                                    Ex. 2115 - pa_ag.v
                                                                                                                                                                               Ex. 2115 - pa_ag.v
 1 u_pntsz_DEBUG
                                                                                                                      .iMEW(pos_mem_we),
2 )
                                                                                                           2
                                                                                                                      .iWEN(pos_mem_we),
     u pntsz dum mem p2 (
                                                                                                                      .iRADR(pos raddr),
                                                                                                                      .iWADR(pos_mem_waddr),
           .iRCLK(sclk).
           iWCLK(sclk)
                                                                                                                      . iD (pos\_pntsz\_ag\_mem\_data),
           .iMER(pntsz re),
                                                                                                                      .oQ(pos_rdata));
           .iMEW(pntsz_mem_we),
           .iWEN(pntsz_mem_we),
                                                                                                           8 // Instantiate the state for ucp,gb,pntsz memory as
           .iRADR(pntsz_raddr),
                                                                                                            9 // 4 64d by 32w to create 64d x 128w
10
           .iWADR(pntsz_mem_waddr),
                                                                                                           10 dum_mem_syn_stub #(
11
           . iD (pos\_pntsz\_ag\_mem\_data [31:0]),
                                                                                                           11 u0_stve_ADDR_WIDTH
12
           .oQ(pntsz\_rdata));
                                                                                                           12 u0_stve_DATA_WIDTH ,
13
                                                                                                           13 u0_stve_WORDS ,
14 // Instantiate the position memory
                                                                                                           14 u0_stve_DEBUG
                                                                                                           15 )
                                                                                                           16 u0_stve_dum_mem_p2 (
16 dum_mem_syn_stub #(
17 u_pos_ADDR_WIDTH,
                                                                                                                      .iRCLK(sclk),
18 u_pos_DATA_WIDTH,
                                                                                                           18
                                                                                                                      .iWCLK(sclk),
19 u_pos_WORDS ,
                                                                                                                      .iMER(stve_re_r0),
20 u pos DEBUG
                                                                                                           20
                                                                                                                     .iMEW(stve_we[0]).
21 )
                                                                                                           21
                                                                                                                     .iWEN(stve_we[0]).
22 u_pos_dum_mem_p2 (
                                                                                                           22
                                                                                                                      .iRADR(stve_raddr).
23
           .iRCLK(sclk),
                                                                                                           23
                                                                                                                      .iWADR(stve wa).
24
           iWCLK(sclk)
                                                                                                           24
                                                                                                                      .iD(rbiu_wd),
25
           .iMER(pos re),
                                                                                                                      .oQ(stve rdata[31:0]));
                                      Page 119 of 123
                                                                                                                                                 Page 120 of 123
                                                                    Ex. 2115 - pa_ag.v
                                                                                                                                                                              Ex. 2115 - pa_ag.v
```

```
PROTECTIVE ORDER MATERIAL
                                                                                                       .iRCLK(sclk),
2 dum_mem_syn_stub #(
                                                                                                       .iWCLK(sclk),
                                                                                                       .iMER(stve_re_r0),
 3 ul_stve_ADDR_WIDTH ,
 4 ul_stve_DATA_WIDTH,
                                                                                                       .iMEW(stve_we[2]),
 5 ul_stve_WORDS ,
                                                                                                       .iWEN(stve_we[2]),
 6 ul_stve_DEBUG
                                                                                                       .iRADR(stve_raddr),
                                                                                                       .iWADR(stve_wa),
 8 u1_stve_dum_mem_p2 (
                                                                                                       .iD(rbiu_wd),
          .iRCLK(sclk),
                                                                                                       .oQ(stve_rdata[95:64]));
         .iWCLK(sclk),
11
         .iMER(stve_re_r0),
                                                                                             11 dum_mem_syn_stub #(
12
        .iMEW(stve_we[1]),
                                                                                             12 u3_stve_ADDR_WIDTH,
13
        .iWEN(stve_we[1]),
                                                                                             13 u3_stve_DATA_WIDTH,
14
        .iRADR(stve_raddr),
                                                                                             14 u3_stve_WORDS ,
15
        .iWADR(stve_wa),
                                                                                             15 u3_stve_DEBUG
16
        .iD(rbiu_wd),
                                                                                             16 )
17
                                                                                             17 u3_stve_dum_mem_p2 (
         .oQ(stve_rdata[63:32]));
18
                                                                                             18
                                                                                                       .iRCLK(sclk).
19 dum_mem_syn_stub #(
                                                                                             19
                                                                                                       .iWCLK(sclk),
20 u2 stve ADDR WIDTH ,
                                                                                             20
                                                                                                      .iMER(stve re r0),
                                                                                                     .iMEW(stve_we[3]),
21 u2_stve_DATA_WIDTH,
                                                                                             21
                                                                                                    .iWEN(stve_we[3]),
22 u2_stve_WORDS ,
                                                                                             22
23 u2_stve_DEBUG
                                                                                             23
                                                                                                      .iRADR(stve_raddr),
24 )
                                                                                             24
                                                                                                       .iWADR(stve_wa),
25 u2_stve_dum_mem_p2 (
                                                                                             25
                                                                                                       .iD(rbiu_wd),
                                 Page 121 of 123
                                                                                                                              Page 122 of 123
                                                           Ex. 2115 - pa_ag.v
                                                                                                                                                        Ex. 2115 - pa_ag.v
          .oQ(stve_rdata[127:96]));
 1
2
 3 'endif
4
 5 endmodule
```

Page 123 of 123

Ex. 2115 - pa_ag.v

```
2 // INPUTS
3 // Project: R400
4 // File: pa_sxifccg.v
                                                                                      4
                                                                                          // global
                                                                                      5 clk.
6 // Description:
                                                                                      8 // ati state storage (sxif)
9 isxif_st_w_data,
                                                                                     10 isxif_st_w_addr,
12 // Trade secret of ATI Technologies, Inc.
                                                                                     11 isxif st we,
                                                                                     12 isxif_st_r_addr,
13 // Copyright 2002, ATI Technologies, Inc., (unpublished)
                                                                                     13 isxif_st_re,
                                                                                     14 isxif_st_sel,
15 // All rights reserved. This notice is intended as a precaution against
                                                                                     15 isxif_st_cpy,
16 // inadvertent publication and does not imply publication or any waiver
17 // of confidentiality. The year included in the foregoing notice is the
                                                                                     17 // vgt_to_ccgen fifo
                                                                                     18 ivgt_to_ccgen_fifo_write,
19 ivgt_to_ccgen_fifo_active_verts,
                                                                                     20 ivgt_to_ccgen_fifo_state_var_indx,
22 'include "header.v"
                                                                                     21
23
                                                                                     22 // sx0, receive
24 module
                                                                                     23 isx0 receive fifo write,
25 pa_sxifccg
                                                                                     24 isx0_receive_fifo_wrdata,
                               Page 1 of 29
                                                                                                                     Page 2 of 29
                                                   Ex. 2116 - pa sxifccg.v
                                                                                                                                         Ex. 2116 - pa sxifccg.v
1 // sx1, receive
                                                                                      1 // state
                                                                                      2 osxif_st_r_data,
2 isx1 receive fifo write,
3 isx1_receive_fifo_wrdata,
                                                                                      4 // state to clipper
5 // ccg state
                                                                                      5 osxif state0,
6 iccg_state0,
                                                                                      6 osxif_state1,
                                                                                      7 osxif_state2,
8 iccg_state2,
                                                                                      8 osxif state3,
                                                                                      9 osxif_state4,
9 iccg state3,
11 iccg_state5,
                                                                                     11 osxif state6,
12 iccg_state6,
                                                                                     12 osxif_state7,
                                                                                     13
14
                                                                                     14 // vgt_to_ccgen fifo
15 // ccgen_to_clipcc/clip
                                                                                     15 ovgt_to_ccgen_fifo_notfull,
17 iccgen_to_clipcc_fifo_full,
                                                                                     17 // sx0, request
                                                                                     18 opa to sx0 req,
18
                                                                                     20 opa_to_sx0_offset,
21
                                                                                     21 opa to sx0 aux,
                                                                                     22 opa_to_sx0_last,
23
24 // OUTPUTS
                                                                                     24 // sx1, request
25 opa_to_sx1_req,
                               Page 3 of 29
                                                                                                                     Page 4 of 29
                                                   Ex. 2116 - pa_sxifccg.v
                                                                                                                                         Ex. 2116 - pa_sxifccg.v
```

ATI 2116 LG v. ATI IPR2015-00326

1	opa_to_sx1_sp_id,		1		
2	opa_to_sxl_offset,		2	// INPUT DECLARATIONS	
3	opa_to_sx1_aux,		3		
4	opa_to_sx1_last,		4	// global	
5			5	input clk;	
6	// position memory		6	input reset;	
7	oposition_write,		7	// ati_state_storage (sxif)	
8	oposition_wraddr,		8	input [SXIF_STATE_WIDTH-1:0] isxif_st_w_data;	
9	oposition_wrdata,		9	input [2:0] isxif_st_w_addr;	
10			10	input isxif_st_we;	
11	// point memory		11	input [2:0] isxif_st_r_addr;	
12	opoint_write,		12	input isxif_st_re;	
13	opoint_wraddr,		13	input isxif_st_sel;	
14	opoint_wrdata,		14	input isxif_st_cpy;	
15	• - /		15	// vgt_to_ccgen fifo	
16	// ccgen_to_clipcc/clip		16	input ivgt_to_ccgen_fifo_write;	
17			17		
	occgen_to_clipec_data,			input [5:0] ivgt_to_ccgen_fifo_active_verts;	
18	occgen_to_clipcc_write,		18	input [2:0] ivgt_to_ccgen_fifo_state_var_indx;	
19	W 15		19	// sx0, receive	
20	// arbiter		20	input isx0_receive_fifo_write;	
21	occgen_to_arb_data		21	input [SX_RECEIVE_FIFO_WIDTH-1:0] isx0_receive_fifo_wr	rdata;
22);		22	// sx1, receive	
23			23	input isx1_receive_fifo_write;	
24	`include "pa_clip_pkg.v"		24	input [SX_RECEIVE_FIFO_WIDTH-1:0] isx1_receive_fifo_wr	rdata;
25			25		
	Page 5 of 29			Page 6 of 29	
	Page 5 of 29	Ex. 2116 - pa_sxifeeg.v		Page 6 of 29	Ex. 2116 - pa_sxifccg.v
	Page 5 of 29	Ex. 2116 - pa_sxifceg.v		Page 6 of 29	Ex. 2116 - pa_sxifceg.v
		Ex. 2116 - pa_sxifceg.v			Ex. 2116 - pa_sxifceg.v
	// ccg state	Ex. 2116 - pa_sxifceg.v	1	output [SXIF_STATE_WIDTH-1:0] osxif_state4;	Ex. 2116 - pa_sxifceg.v
2	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0;	Ex. 2116 - pa_sxifceg.v	1 2	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5;	Ex. 2116 - pa_sxifceg.v
	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1;	Ex. 2116 - pa_sxifceg.v	1	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6;	Ex. 2116 - pa_sxifceg.v
2	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2;	Ex. 2116 - pa_sxifceg.v	1 2	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5;	Ex. 2116 - pa_sxifceg.v
2	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3;	Ex. 2116 - pa_sxifceg.v	1 2 3	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 5	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull;	Ex. 2116 - pa_sxifceg.v
2 3 4 5	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 5	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 5	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 5 6	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 5 6 7	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state6;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 5 6 7 8	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 9	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip	Ex. 2116 - pa_sxifceg.v	1 2 3 3 4 4 5 6 6 7 8 8 9 10	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovet_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 9 10	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices;	Ex. 2116 - pa_sxifceg.v	1 2 3 3 4 4 5 6 6 7 8 8 9 10 11	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 9 10 11	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full;	Ex. 2116 - pa_sxifceg.v	1 2 3 3 4 4 5 5 6 6 7 8 8 9 10 11 12 12	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 9 10 11 12 13	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter	Ex. 2116 - pa_sxifceg.v	1 2 3 3 4 4 5 5 6 6 7 7 8 9 10 11 12 13	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request	Ex. 2116 - pa_sxifceg.v
2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter	Ex. 2116 - pa_sxifceg.v	1 2 3 3 4 4 5 5 6 6 7 7 8 9 10 11 12 13 14	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_req; output opa_to_sx0_sx_offset; output [1:0] opa_to_sx_Oaux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_req;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 8 9 10 11 12 13 14 15	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_req; output opa_to_sx0_offset; output [1:0] opa_to_sx0_offset; output opa_to_sx0_last; // sx1, request output opa_to_sx1_req; output opa_to_sx1_req; output opa_to_sx1_sp_id;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_req; output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_sp_id; output [1:0] opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output [1:0] opa_to_sx1_offset; output [1:0] opa_to_sx1_aux;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 6 7 8 9 10 11 12 13 13 14 15 16 17	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_req; output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_aux; output opa_to_sx1_aux; output opa_to_sx1_aux; output opa_to_sx1_last;	Ex. 2116 - pa_sxifceg.v
2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 13 14 15 16 17 18 19	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipcc_fifo_full; // arbiter input iarb_to_ccgen_xfc;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_cegen fifo output ovgt_to_cegen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_eq; output opa_to_sx1_offset; output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_aux; output opa_to_sx1_last; // position memory	Ex. 2116 - pa_sxifceg.v
2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce fifo_full; // arbiter input iarb_to_ccgen_xfc; // OUTPUT DECLARATIONS	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_cegen fifo output ovgt_to_cegen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_last; // position memory output oposition_write;	Ex. 2116 - pa_sxifceg.v
2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipcc_fifo_full; // arbiter input iarb_to_ccgen_xfc; // OUTPUT DECLARATIONS	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_aux; output opa_to_sx1_aux; output opa_to_sx1_last; // position memory output oposition_write; output [5:0] oposition_wraddr;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipcc/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipcc_fifo_full; // arbiter input iarb_to_ccgen_xfc; /// OUTPUT DECLARATIONS /// state output [SXIF_STATE_WIDTH-1:0] osxif_st_r_data; // state to clipper output [SXIF_STATE_WIDTH-1:0] osxif_state0;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_aux; output opa_to_sx1_aux; output opa_to_sx1_last; // position memory output [5:0] oposition_wraddr; output [1:7:0] oposition_wraddr; output [1:7:0] oposition_wradda;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc; /// OUTPUT DECLARATIONS /// state output [SXIF_STATE_WIDTH-1:0] osxif_st_r_data; // state to clipper output [SXIF_STATE_WIDTH-1:0] osxif_state0; output [SXIF_STATE_WIDTH-1:0] osxif_state1;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_laux; output opa_to_sx1_laux; output opa_to_sx1_laux; output oposition_write; output [5:0] oposition_wraddr; output [127:0] oposition_wraddat; // point memory	Ex. 2116 - pa_sxifceg.v
2 3 3 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc; /// OUTPUT DECLARATIONS /// state output [SXIF_STATE_WIDTH-1:0] osxif_st_r_data; // state o clipper output [SXIF_STATE_WIDTH-1:0] osxif_state0; output [SXIF_STATE_WIDTH-1:0] osxif_state1; output [SXIF_STATE_WIDTH-1:0] osxif_state2;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_aux; output opa_to_sx1_last; // position memory output [1:7:0] oposition_wraddr; output [1:7:0] oposition_wradta; // point memory output opoint_write;	Ex. 2116 - pa_sxifceg.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc; /// OUTPUT DECLARATIONS /// state output [SXIF_STATE_WIDTH-1:0] osxif_st_r_data; // state to clipper output [SXIF_STATE_WIDTH-1:0] osxif_state0; output [SXIF_STATE_WIDTH-1:0] osxif_state1;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_laux; output opa_to_sx1_laux; output opa_to_sx1_laux; output oposition_write; output [5:0] oposition_wraddr; output [127:0] oposition_wraddat; // point memory	Ex. 2116 - pa_sxifceg.v
2 3 3 4 5 6 6 7 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// ccg state input [CCG_STATE_WIDTH-1:0] iccg_state0; input [CCG_STATE_WIDTH-1:0] iccg_state1; input [CCG_STATE_WIDTH-1:0] iccg_state2; input [CCG_STATE_WIDTH-1:0] iccg_state3; input [CCG_STATE_WIDTH-1:0] iccg_state4; input [CCG_STATE_WIDTH-1:0] iccg_state5; input [CCG_STATE_WIDTH-1:0] iccg_state6; input [CCG_STATE_WIDTH-1:0] iccg_state7; // ccgen_to_clipce/clip input [1:0] ioutsm_clr_orig_vertices; input iccgen_to_clipce_fifo_full; // arbiter input iarb_to_ccgen_xfc; /// OUTPUT DECLARATIONS /// state output [SXIF_STATE_WIDTH-1:0] osxif_st_r_data; // state o clipper output [SXIF_STATE_WIDTH-1:0] osxif_state0; output [SXIF_STATE_WIDTH-1:0] osxif_state1; output [SXIF_STATE_WIDTH-1:0] osxif_state2;	Ex. 2116 - pa_sxifceg.v	1 2 3 4 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	output [SXIF_STATE_WIDTH-1:0] osxif_state4; output [SXIF_STATE_WIDTH-1:0] osxif_state5; output [SXIF_STATE_WIDTH-1:0] osxif_state6; output [SXIF_STATE_WIDTH-1:0] osxif_state7; // vgt_to_ccgen_fifo output ovgt_to_ccgen_fifo_notfull; // sx0, request output opa_to_sx0_req; output opa_to_sx0_sp_id; output [1:0] opa_to_sx0_offset; output opa_to_sx0_aux; output opa_to_sx0_last; // sx1, request output opa_to_sx1_sp_id; output [1:0] opa_to_sx1_offset; output opa_to_sx1_aux; output opa_to_sx1_last; // position memory output [1:7:0] oposition_wraddr; output [1:7:0] oposition_wradta; // point memory output opoint_write;	Ex. 2116 - pa_sxifceg.v

1	output [31:0] opoint_wrdata;	1 reg sx0_receive_fifo_write;
2		2 reg [SX_RECEIVE_FIFO_WIDTH-1:0] sx0_receive_fifo_wrdata;
3	// ccgen_to_clipcc/clip	3 // sx1, receive
4	output [CCGEN_TO_CLIPCC_FIFO_WIDTH-1:0] occgen_to_clipcc_data;	4 reg sx1_receive_fifo_write;
5	output occgen_to_clipcc_write;	5 reg [SX_RECEIVE_FIFO_WIDTH-1:0] sx1_receive_fifo_wrdata;
6	// arbiter	6
7	output [15:0] occgen_to_arb_data;	7 // ccg state
8		8 reg [CCG_STATE_WIDTH-1:0] ccg_state0;
9		9 reg [CCG_STATE_WIDTH-1:0] ccg_state1;
10		10 reg [CCG_STATE_WIDTH-1:0] ccg_state2;
11	// SIGNAL DECLARATIONS	11 reg [CCG_STATE_WIDTH-1:0] ccg_state3;
12		12 reg [CCG_STATE_WIDTH-1:0] ccg_state4;
13	// INPUTS	13 reg [CCG_STATE_WIDTH-1:0] ccg_state5;
14	// ati_state_storage (sxif)	14 reg [CCG_STATE_WIDTH-1:0] ccg_state6;
15	reg [SXIF_STATE_WIDTH-1:0] sxif_st_w_data;	15 reg [CCG_STATE_WIDTH-1:0] ccg_state7;
16	reg [2:0] sxif_st_w_addr;	16 // ccgen_to_clipcc/clip
17	reg sxif_st_we;	17 reg [1:0] outsm_clr_orig_vertices;
18	reg [2:0] sxif_st_r_addr;	18 reg ccgen_to_clipcc_fifo_full;
19	reg sxif_st_re;	19 // arbiter
20	reg sxif_st_sel;	20 reg arb_to_ccgen_xfc;
21	reg sxif_st_cpy;	21
22	// vgt_to_ccgen fifo	22 // OUTPUTS
23	reg vgt_to_ccgen_fifo_write;	23 // state
24	reg [VGT_TO_CCGEN_FIFO_WIDTH-1:0] vgt_to_ccgen_fifo_wrdata;	24 wire [SXIF_STATE_WIDTH-1:0] sxif_st_r_data;
25	// sx0, receive	25 // vertex_fifo (debug)
	Page 9 of 29	D 10 . C20
		Page 10 of 29
	Ex. 2116 - pa_sxifceg v	Page 10 of 29 Ex. 2116 - pa_sxifccg.v
		Ex. 2116 - pa_sxifceg.v
1	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata;	Ex. 2116 - pa_sxifceg.v
2	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write;	Ex. 2116 - pa_sxifceg.v 1 2 // cegen_to_clipce/clip
2	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo	Ex. 2116 - pa_sxifceg.v 1 2 // cegen_to_clipce/clip 3 wire [CCGEN_TO_CLIPCC_FIFO_WIDTH-1:0] cegen_to_clipce_data;
2 3 4	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full;	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen_fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full;	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6 7	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write;	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata;	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6 7 8 9	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6 7 8 9 10	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive_fifo_full;	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6 7 8 9 10 11 12	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive_fifo_full; // sx1, request	Ex. 2116 - pa_sxifccg.v 1
2 3 4 5 6 7 8 9 10 11 12 13	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write;	Ex. 2116 - pa_sxifccg.v 1 2
2 3 4 5 6 7 8 9 10 11 12 13	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_write;	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8 9 10 11 12 13	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write;	Ex. 2116 - pa_sxifccg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, receive wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr;	Ex. 2116 - pa_sxifccg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write;	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, receive wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr; wire [127:0] position_wrdata; // point memory	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, receive wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr; wire [127:0] position_wrdata; // point memory wire point_write;	Ex. 2116 - pa_sxifccg.v 1
2 3 4 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr; wire [127:0] position_wrdata; // point memory wire point_write; wire [5:0] point_wraddr;	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr; wire [127:0] position_wrdata; // point memory wire point_write; wire [5:0] point_wrdadr; wire [5:0] point_wrdadr; wire [5:0] point_wraddr;	Ex. 2116 - pa_sxifecg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wrdata; // point memory wire point_write; wire [5:0] point_wraddr; wire [31:0] point_wradata; // sx_pending_fifo (debug)	Ex. 2116 - pa_sxifceg.v 1 2
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr; wire [127:0] position_wrdata; // point memory wire point_write; wire [5:0] point_wrdadr; wire [5:0] point_wrdadr; wire [5:0] point_wraddr;	Ex. 2116 - pa_sxifceg.v 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wraddr; wire [127:0] position_wrdata; // point memory wire point_write; wire [5:0] point_wraddr; wire [17:0] position_wrdata; // sx_pending_fifo_debug) wire sx_pending_fifo_write; wire [17:0] sx_pending_fifo_wrdata;	Ex. 2116 - pa_sxifeegv 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	wire [VERTEX_FIFO_WIDTH-1:0] vertex_fifo_wrdata; wire vertex_fifo_write; // vgt_to_ccgen fifo wire vgt_to_ccgen_fifo_full; // sx0, receive wire sx0_receive_fifo_full; // sx0, request wire pa_to_sx0_write; wire [5:0] pa_to_sx0_wrdata; // sx1, receive wire sx1_receive_fifo_full; // sx1, request wire pa_to_sx1_write; wire [5:0] pa_to_sx1_wrdata; // position memory wire position_write; wire [5:0] position_wrdata; // point memory wire position_write; wire [5:0] position_wrdata; // point memory wire [5:0] point_wrdata; // sx_pending_fifo (debug) wire sx_pending_fifo (write;	Ex. 2116 - pa_sxifceg.v

e vertex_fifo_advanceread; e [6:0] available_positions; e decrement_available_positions; e [SXIF_STATE_WIDTH-1:0] sxif_st_data0; e [SXIF_STATE_WIDTH-1:0] sxif_st_data1; e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; MAP INPUTS MAP INPUTS	l ivgt_to_ccgen_fifo_write or livgt_to_ccgen_fifo_active_verts or livgt_to_ccgen_fifo_state_var_indx or livgt_to_ccgen_fifo_state_var_indx or livgt_to_ccgen_fifo_state_var_indx or livgt_to_ccgen_fifo_state_var_indx or livgt_to_ccgen_fifo_write or l
e [6.0] available_positions; e decrement_available_positions; e [SXIF_STATE_WIDTH-1:0] sxif_st_data0; e [SXIF_STATE_WIDTH-1:0] sxif_st_data1; e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; MAP INPUTS	2 ivgt_to_ccgen_fifo_active_verts or 3 ivgt_to_ccgen_fifo_state_var_indx or 4 // sx0 5 isx0_receive_fifo_write or 6 isx0_receive_fifo_wrdata or 7 // sx1 8 isx1_receive_fifo_wrdata or 9 isx1_receive_fifo_wrdata or 10 // vertex fifo/ccg 11 12 // ccg_state 13 iccg_state0 or
e [6.0] available_positions; e decrement_available_positions; e [SXIF_STATE_WIDTH-1:0] sxif_st_data0; e [SXIF_STATE_WIDTH-1:0] sxif_st_data1; e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; MAP INPUTS	2 ivgt_to_ccgen_fifo_active_verts or 3 ivgt_to_ccgen_fifo_state_var_indx or 4 // sx0 5 isx0_receive_fifo_write or 6 isx0_receive_fifo_wrdata or 7 // sx1 8 isx1_receive_fifo_wrdata or 9 isx1_receive_fifo_wrdata or 10 // vertex fifo/ccg 11 12 // ccg_state 13 iccg_state0 or
e decrement_available_positions; e [SXIF_STATE_WIDTH-1:0] sxif_st_data0; e [SXIF_STATE_WIDTH-1:0] sxif_st_data1; e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; ###################################	3 ivgt_to_ccgen_fifo_state_var_indx or 4
e [SXIF_STATE_WIDTH-1:0] sxif_st_data0; e [SXIF_STATE_WIDTH-1:0] sxif_st_data1; e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; MAP INPUTS	4
e [SXIF_STATE_WIDTH-1:0] sxif_st_data1; e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7;	5 isx0_receive_fifo_write or 6 isx0_receive_fifo_wrdata or 7
e [SXIF_STATE_WIDTH-1:0] sxif_st_data2; e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7;	6 isx0_receive_fifo_wrdata or 7
e [SXIF_STATE_WIDTH-1:0] sxif_st_data3; e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7;	7
e [SXIF_STATE_WIDTH-1:0] sxif_st_data4; e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7;	8 isx1_receive_fifo_write or 9 isx1_receive_fifo_wrdata or 10
e [SXIF_STATE_WIDTH-1:0] sxif_st_data5; e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; MAP INPUTS	9 isx1_receive_fifo_wrdata or 10 // vertex fifo/ccg 11 12 // ccg state 13 iccg_state0 or
e [SXIF_STATE_WIDTH-1:0] sxif_st_data6; e [SXIF_STATE_WIDTH-1:0] sxif_st_data7; IAP INPUTS	10 // vertex fifo/ccg 11 12 // ccg state 13 iccg_state0 or
e [SXIF_STATE_WIDTH-1:0] sxif_st_data7;	11 12 // ccg state 13 iccg_state0 or
tap inputs	12 // ccg state 13 iccg_state0 or
IAP INPUTS	13 iceg_state0 or
IAP INPUTS	
ays (a)(15 iccg_state2 or
	16 iccg_state3 or
// ati_state_storage (sxif)	17 iccg_state4 or 18 iccg_state5 or
isxif_st_w_data or	
isxif_st_w_addr or isxif_st_we or	19 iccg_state6 or 20 iccg_state7 or
	20 1ccg_state/ 01
isxif_st_r_addr or isxif_st_re or	22 // ccgen_to_clipce/clip
isxif_st_sel or	23 ioutsm_clr_orig_vertices or
isxif_st_cpy or	24 iccgen_to_clipce_fifo_full or
// vgt_to_ccgen fifo	25
Page 13 of 29 Ex. 2116 - pa_sxifceg.v	Page 14 of 29 Ex. 2116 - pa_sxifccg.v
// arbiter iarb_to_ccgen_xfe) begin : proc000	1
ati_state_storage	4 ccg_state4 = iccg_state4;
if_st_w_data = isxif_st_w_data;	5 ccg_state5 = iccg_state5;
if_st_w_addr = isxif_st_w_addr;	6 ccg_state6 = iccg_state6;
$if_st_we = isxif_st_we;$	7 ccg_state7 = iccg_state7;
if_st_r_addr = isxif_st_r_addr;	8
if_st_re = isxif_st_re;	9 // ccgen_to_clipcc/clip
if at ant - innif at ant.	10 outsm_clr_orig_vertices = ioutsm_clr_orig_vertices;
if_st_sel = isxif_st_sel;	11 ccgen_to_clipce_fifo_full = iccgen_to_clipce_fifo_full; 12
if_st_sel = isxif_st_sel; if_st_cpy = isxif_st_cpy;	12
if_st_cpy = isxif_st_cpy;	13 // orbitor
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo	13 // arbiter
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write;	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc;
<pre>if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,</pre>	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts, ivgt_to_ccgen_fifo_state_var_indx};	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16
<pre>if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,</pre>	14 arb_to_ccgen_xfe = iarb_to_ccgen_xfe; 15 end 16
<pre>if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,</pre>	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16
<pre>if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,</pre>	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,	14 arb_to_ccgen_xfe = iarb_to_ccgen_xfe; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,	14 arb_to_ccgen_xfe = iarb_to_ccgen_xfe; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy = isxif_st_cpy; vgt_to_ccgen fifo t_to_ccgen_fifo_write = ivgt_to_ccgen_fifo_write; t_to_ccgen_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,	14 arb_to_ccgen_xfc = iarb_to_ccgen_xfc; 15 end 16 17 18 ////////////////////////////////////
if_st_cpy vgt_to_c t_to_ccg	ten_fifo_write = ivgt_to_ccgen_fifo_write; ten_fifo_wrdata = {ivgt_to_ccgen_fifo_active_verts,

```
1 assign osxif state1 = sxif st data1;
                                                                                                    1 assign oposition_wrdata = position_wrdata;
2 assign osxif state2 = sxif st data2;
                                                                                                    2 // point memory
 3 assign osxif state3 = sxif st data3;
                                                                                                    3 assign opoint_write = point_write;
 4 assign osxif_state4 = sxif_st_data4;
                                                                                                    4 assign opoint_wraddr = point_wraddr;
     assign\ osxif\_state5 = sxif\_st\_data5;
                                                                                                    5 assign opoint_wrdata = point_wrdata;
     assign osxif_state6 = sxif_st_data6;
     assign osxif_state7 = sxif_st_data7;
                                                                                                    7 // ccgen_to_clipcc/clip
                                                                                                    8 assign occgen_to_clipcc_data = ccgen_to_clipcc_data;
 9 // vgt_to_ccgen fifo
                                                                                                    9 assign occgen_to_clipcc_write = ccgen_to_clipcc_write;
10 \quad assign\ ovgt\_to\_ccgen\_fifo\_notfull = \sim vgt\_to\_ccgen\_fifo\_full;
11 // sx0, request
                                                                                                   11 assign occgen_to_arb_data = ccgen_to_arb_data;
12 assign {opa_to_sx0_req,
13
        opa_to_sx0_sp_id,
                                                                                                   13
        opa_to_sx0_offset,
                                                                                                    15
        opa to sx0 aux,
                                                                                                   15 // vgt to ccgen fifo
                                                                                                   16
        opa to sx0 last} = pa to sx0 wrdata;
17 // sx1. request
                                                                                                   17 ati_fifo_top
18 assign {opa_to_sx1_req,
                                                                                                   18 #(
19
       opa to sxl sp id,
                                                                                                   19
                                                                                                        VGT TO CCGEN FIFO WIDTH,
                                                                                                        VGT TO CCGEN FIFO POINTER SIZE,
20
       opa to sx1 offset,
                                                                                                   20
                                                                                                        VGT TO CCGEN FIFO DEPTH.
21
        opa to sx1 aux,
22
       opa_to_sx1_last} = pa_to_sx1_wrdata;
                                                                                                   22
                                                                                                        VGT_TO_CCGEN_FIFO_SKID_WORDS
23 // position memory
                                                                                                   23
24 assign oposition_write = position_write;
                                                                                                   24 uvgt_to_ccgen_fifo
25 assign oposition_wraddr = position_wraddr;
                                                                                                   25
                                    Page 17 of 29
                                                                                                                                        Page 18 of 29
                                                           Ex. 2116 - pa_sxifccg.v
                                                                                                                                                               Ex. 2116 - pa_sxifccg.v
                                                                                                              (sx0_receive_fifo_write),
     .write data (vgt to ccgen fifo wrdata),
2
                                                                                                    2 .full (sx0 receive fifo full),
     .we (vgt to ccgen fifo write),
     .full (vgt to ccgen fifo full),
                                                                                                         .read data (sx0 receive fifo rddata),
     .read data (vgt to ccgen fifo rddata),
                                                                                                         .empty (sx0 receive fifo empty),
     .empty (vgt_to_ccgen_fifo_empty),
                                                                                                         .busy (sx0_receive_fifo_busy_nc),
     .busy (vgt_to_ccgen_fifo_busy_nc),
                                                                                                              (sx0_receive_fifo_advanceread)
     .re (vgt_to_ccgen_fifo_advanceread),
                                                                                                         clk
                                                                                                               (clk).
     .clk (clk).
                                                                                                         .reset (reset)
     .reset (reset)
                                                                                                    9
10
                                                                                                   10
11
                                                                                                   11
12
                                                                                                   13 // sx1 receive fifo
14 // sx0 receive fifo
                                                                                                   15 ati_fifo_top
17 #(
                                                                                                   17 SX_RECEIVE_FIFO_WIDTH,
18
     SX_RECEIVE_FIFO_WIDTH,
                                                                                                   18 SX_RECEIVE_FIFO_POINTER_SIZE,
19 SX_RECEIVE_FIFO_POINTER_SIZE,
                                                                                                   19 SX_RECEIVE_FIFO_DEPTH,
     SX_RECEIVE_FIFO_DEPTH,
                                                                                                   20 SX_RECEIVE_FIFO_SKID_WORDS
20
21
    SX_RECEIVE_FIFO_SKID_WORDS
                                                                                                   21 )
22 )
                                                                                                   22 usx1_receive_fifo
23 usx0 receive fifo
                                                                                                   23 (
24 (
                                                                                                   24 .write_data (sx1_receive_fifo_wrdata),
     .write_data (sx0_receive_fifo_wrdata)
                                                                                                        .we (sx1_receive_fifo_write),
                                    Page 19 of 29
                                                                                                                                        Page 20 of 29
                                                           Ex. 2116 - pa_sxifccg.v
                                                                                                                                                               Ex. 2116 - pa_sxifccg.v
```

```
.full (sx1 receive fifo full).
                                                                                                     .read_data (vertex_fifo_rddata),
2
     .read_data (sx1_receive_fifo_rddata),
                                                                                                     .empty (vertex_fifo_empty),
      .empty (sx1_receive_fifo_empty),
                                                                                                     .busy (vertex_fifo_busy_nc),
      .busy (sxl_receive_fifo_busy_nc),
                                                                                                           (vertex_fifo_advanceread),
     .re (sx1_receive_fifo_advanceread),
10
                                                                                                   // ati_8rp_state_storage
11
    12
   // vertex_fifo fifo
                                                                                                12 ati_8rp_state_storage
13
    13 #(
                                                                                                     SXIF_STATE_WIDTH,
14
   ati_fifo_top
15 #(
                                                                                                15
                                                                                                     SXIF_STATES
     VERTEX_FIFO_WIDTH,
                                                                                                16
16
17
     VERTEX FIFO POINTER SIZE,
                                                                                                17 uati_8rp_state_storage
     VERTEX FIFO DEPTH,
                                                                                                18 (
18
19
     VERTEX FIFO SKID WORDS
                                                                                                19
                                                                                                     .st_data0 (sxif_st_data0),
20 )
                                                                                                20
                                                                                                     .st data1 (sxif st data1),
21 uvertex fifo
                                                                                                     .st_data2 (sxif_st_data2),
                                                                                                21
22 (
                                                                                                22
                                                                                                     .st_data3 (sxif_st_data3),
23
     .write_data (vertex_fifo_wrdata),
                                                                                                23
                                                                                                     .st_data4 (sxif_st_data4),
24
     .we (vertex_fifo_write),
                                                                                                24
                                                                                                     .st_data5 (sxif_st_data5),
     .full (vertex_fifo_full),
                                                                                                     .st_data6 (sxif_st_data6),
                                   Page 21 of 29
                                                                                                                                   Page 22 of 29
                                                         Ex. 2116 - pa_sxifccg.v
                                                                                                                                                         Ex. 2116 - pa_sxifccg.v
     .st_data7 (sxif_st_data7),
                                                                                                     r data (sxif st r data),
      .w_data (sxif_st_w_data),
                                                                                                     // INPUTS
      .w addr (sxif st w addr),
                                                                                                     .r_addr (sxif_st_r_addr),
                                                                                                     // vgt_to_ccgen fifo
             (sxif_st_we),
                                                                                                     . ivgt\_to\_ccgen\_fifo\_empty \\ \hspace*{0.5cm} (vgt\_to\_ccgen\_fifo\_empty), \\
            (sxif_st_re),
                                                                                                     .ivgt_to_ccgen_fifo_rddata
                                                                                                                              (vgt_to_ccgen_fifo_rddata),
            (sxif_st_sel),
            (sxif_st_cpy),
                                                                                                     // state
10
      .clk
            (clk)
                                                                                                10
                                                                                                     .isxif state0
                                                                                                                          (sxif_st_data0),
11
                                                                                                     .isxif_state1
                                                                                                                          (sxif_st_data1),
12
                                                                                                     .isxif_state2
                                                                                                                          (sxif_st_data2),
13
                                                                                                                          (sxif_st_data3),
14
                                                                                                                          (sxif_st_data4),
    .isxif_state5
                                                                                                                          (sxif_st_data5),
                                                                                                     .isxif_state6
                                                                                                                          (sxif_st_data6),
    // sxif state machine
17
    .isxif_state7
                                                                                                                          (sxif_st_data7),
18
   pa ccg sxifsm
                                                                                                18
19
                                                                                                     // vertex_fifo
    upa ccg sxifsm
20
                                                                                                20
                                                                                                     ivertex fifo full
                                                                                                                           (vertex fifo full),
21
     21
                                                                                                     .ivertex fifo advanceread
                                                                                                                             (vertex fifo advanceread),
22
                                                                                                22
    // GLOBAL SIGNALS
23
     23
                                                                                                     // ccg
               (clk),
24
                                                                                                24
                                                                                                     .iavailable_positions
                                                                                                                            (available_positions),
     reset
                      (reset),
                                   Page 23 of 29
                                                                                                                                   Page 24 of 29
                                                         Ex. 2116 - pa_sxifccg.v
                                                                                                                                                         Ex. 2116 - pa_sxifccg.v
```

```
1 // sx to pa 0
                                                                                                          .omem_position_wraddr
                                                                                                                                     (position_wraddr),
2
                              (sx0_receive_fifo_empty),
     .isx_to_pa_empty_0
                                                                                                          .omem_position_wrdata
                                                                                                                                     (position_wrdata),
      .isx_to_pa_vector_0
                              (sx0_receive_fifo_rddata),
                                                                                                      4 // point memory
     // sx to pa 1
                                                                                                          .omem_point_write
                                                                                                                                   (point write),
      .isx_to_pa_empty_1
                               (sx1_receive_fifo_empty),
                                                                                                          .omem_point_wraddr
                                                                                                                                    (point_wraddr),
      .isx_to_pa_vector_1
                              (sx1_receive_fifo_rddata),
                                                                                                          .omem_point_wrdata
                                                                                                                                    (point_wrdata),
      . odecrement\_available\_positions \qquad (decrement\_available\_positions),
11
12
13
     // vgt_to_ccgen fifo
                                                                                                        .overtex_fifo_write
                                                                                                                               (vertex_fifo_write),
14
      .ovgt_to_ccgen_fifo_advanceread (vgt_to_ccgen_fifo_advanceread),
                                                                                                         .overtex_fifo_wrdata
                                                                                                                                 (vertex_fifo_wrdata),
15
                                                                                                     15
16
                                                                                                          // sx_to_pa_0
     // pa to sx0
                                                                                                     16
17
      .opa_to_sx0_write
                           (pa_to_sx0_write),
                                                                                                    17
                                                                                                                                     (sx0_receive_fifo_advanceread),
                                                                                                          .osx_to_pa_advanceread_0
                                                                                                     18
18
      .opa_to_sx0_wrdata
                             (pa to sx0 wrdata),
19
                                                                                                     19 // sx to_pa_1
                                                                                                          .osx_to_pa_advanceread_1
20
     // na to sx1
                                                                                                    20
                                                                                                                                     (sx1 receive fifo advanceread),
     .opa_to_sx1_write
21
                             (pa_to_sx1_write),
                                                                                                    21
                                                                                                    22 // debug only
22
      .opa\_to\_sx\,l\_wrdata
                              (pa_to_sx1_wrdata),
23
                                                                                                    23
                                                                                                          .osx_pending_fifo_write
                                                                                                                                     (sx_pending_fifo_write),
24
     // position memory
                                                                                                     24
                                                                                                          .osx_pending_fifo_wrdata
                                                                                                                                     (sx_pending_fifo_wrdata)
      .omem_position_write
                               (position_write),
                                                                                                     25
                                     Page 25 of 29
                                                                                                                                          Page 26 of 29
                                                            Ex. 2116 - pa_sxifccg.v
                                                                                                                                                                 Ex. 2116 - pa_sxifccg.v
                                                                                                          .iccg state6
                                                                                                                                (ccg state6),
2
                                                                                                          .iccg state7
                                                                                                                                (ccg_state7),
     pa_ccg_ccgsm
 4 upa_ccg_ccgsm
                                                                                                          // clipper
                                                                                                          .ioutsm_clr_orig_vertices
                                                                                                                                   (outsm_clr_orig_vertices),
     // COMMON
                                                                                                         // arbiter
      .iarb_to_ccgen_xfc
                                                                                                                                   (arb_to_ccgen_xfc),
                     (clk),
10
                       (reset),
                                                                                                     10
                                                                                                          // ccgen_to_clipcc
11
                                                                                                    11
                                                                                                          .iccgen_to_clipcc_fifo_full
                                                                                                                                     (ccgen_to_clipcc_fifo_full),
12
13 // INPUTS
14
     .idecrement_available_positions (decrement_available_positions),
15
                           (vertex_fifo_empty),
                                                                                                     . ivertex\_fifo\_empty
17
      .ivertex_fifo_rd_data
                          (vertex_fifo_rddata),
18
                                                                                                     19
     // state
                                                                                                          .overtex_fifo_advanceread (vertex_fifo_advanceread),
20
     .iccg state0
                           (ccg_state0).
                                                                                                     20
21
                                                                                                    21
     .iccg state1
                           (ccg state1).
      .iccg_state2
                           (ccg_state2),
22
                                                                                                     22 // ccgen_to_clipcc
23
      .iccg_state3
                                                                                                     23 .occgen_to_clipcc_data
                           (ccg_state3),
                                                                                                                                    (ccgen to clipcc data),
24
      .iccg_state4
                           (ccg_state4),
                                                                                                     24
                                                                                                          .occgen_to_clipcc_write
                                                                                                                                    (ccgen_to_clipcc_write),
      .iccg_state5
                           (ccg_state5),
                                    Page 27 of 29
                                                                                                                                          Page 28 of 29
                                                            Ex. 2116 - pa_sxifccg.v
                                                                                                                                                                 Ex. 2116 - pa_sxifccg.v
```

```
1  // sxif
2  .oavailable_positions (available_positions),
3
4  // arbiter
5  .oecgen_to_arb_data (ecgen_to_arb_data)
6  );
7
8
9
10  endmodule
11
12

Page 29 of 29

Ex. 2116 - pa_sxifeeg_v
```

```
2 // GLOBAL SIGNALS
3 // Project: R400
4 // File: pa_ccg_sxifsm.v
6 // Description:
                                                                                   8 // INPUTS
10 // vgt_to_ccgen fifo
12 // Trade secret of ATI Technologies, Inc.
                                                                                  11 ivgt_to_ccgen_fifo_empty,
                                                                                  12 ivgt_to_ccgen_fifo_rddata,
13 // Copyright 2002, ATI Technologies, Inc., (unpublished)
                                                                                  13
                                                                                  14 // state
15 // All rights reserved. This notice is intended as a precaution against
                                                                                  15 isxif_state0,
16 // inadvertent publication and does not imply publication or any waiver
                                                                                  16 isxif_state1,
17 // of confidentiality. The year included in the foregoing notice is the
                                                                                  17 isxif state2.
                                                                                  18 isxif_state3,
20 isxif_state5,
22 'include "header.v"
                                                                                  21 isxif state6.
23
                                                                                  22 isxif_state7,
24 module
                                                                                  23
25 pa_ccg_sxifsm
                                                                                  24 // vertex_fifo
                                                                                  25 ivertex_fifo_full,
                              Page 1 of 48
                                                                                                                 Page 2 of 48
                                              Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                 Ex. 2117 - pa_ccg_sxifsm.v

    ivertex_fifo_advanceread,

                                                                                   1 opa_to_sx1_write,
                                                                                   2 opa_to_sx1_wrdata,
3 // ccg
4 iavailable_positions,
                                                                                   5 omem position write,
6 // sx_to_pa_0
                                                                                   6 omem_position_wraddr,
                                                                                   7 omem_position_wrdata,
8 isx_to_pa_vector_0,
                                                                                   9 // point memory
11 isx_to_pa_empty_1,
                                                                                  11 omem_point_wraddr,
12 isx_to_pa_vector_1,
                                                                                  12 omem_point_wrdata,
13
                                                                                  13
14
                                                                                  14 // ccg
15 odecrement_available_positions,
17 // vertex fifo
18 // vgt to ccgen fifo
                                                                                  18 overtex fifo write,
                                                                                  19 overtex_fifo_wrdata,
19 ovgt_to_ccgen_fifo_advanceread,
20
                                                                                  20
21 // pa_to_sx0
                                                                                  21 // sx_to_pa_0
22 opa_to_sx0_write,
                                                                                  22 osx_to_pa_advanceread_0,
23 opa_to_sx0_wrdata,
                                                                                  23
                                                                                  24 // sx_to_pa_1
25 // pa_to_sx1
                                                                                  25 osx_to_pa_advanceread_1,
                              Page 3 of 48
                                                                                                                 Page 4 of 48
                                              Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                 Ex. 2117 - pa_ccg_sxifsm.v
```

ATI 2117 LG v. ATI IPR2015-00326

```
1 input [SXIF_STATE_WIDTH-1:0] isxif_state5;
2
                                                                                                           2 input [SXIF_STATE_WIDTH-1:0] isxif_state6;
     // debug only
                                                                                                              input [SXIF_STATE_WIDTH-1:0] isxif_state7;
      osx_pending_fifo_write,
      osx_pending_fifo_wrdata
                                                                                                           5 // vertex fifo
                                                                                                           6 input ivertex_fifo_full;
     'include "pa_clip_pkg.v"
                                                                                                              input ivertex_fifo_advanceread;
    // INPUT DECLARATIONS
                                                                                                              input [6:0] iavailable_positions;
                                                                                                          12 // sx_to_pa_0
13 input clk;
                                                                                                          13 input isx_to_pa_empty_0;
                                                                                                          14 input [127:0] isx_to_pa_vector_0;
     input reset;
15
                                                                                                          15
    // vgt to ccgen fifo
                                                                                                          16 // sx to pa 1
16
17
                                                                                                          17 input isx_to_pa_empty_1;
     input ivgt to ccgen fifo empty;
     input [VGT_TO_CCGEN_FIFO_WIDTH-1:0] ivgt_to_ccgen_fifo_rddata;
                                                                                                          18 input [127:0] isx_to_pa_vector_1;
18
19
                                                                                                          19
20
                                                                                                          input [SXIF_STATE_WIDTH-1:0] isxif_state0;
                                                                                                          21 // OUTPUT DECLARATIONS
21
    input [SXIF_STATE_WIDTH-1:0] isxif_state1;
                                                                                                          input [SXIF_STATE_WIDTH-1:0] isxif_state2;
                                                                                                          23 // vgt_to_ccgen fifo
     input [SXIF_STATE_WIDTH-1:0] isxif_state3;
                                                                                                              output ovgt_to_ccgen_fifo_advanceread;
     input [SXIF_STATE_WIDTH-1:0] isxif_state4;
                                       Page 5 of 48
                                                                                                                                                 Page 6 of 48
                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                     Ex. 2117 - pa_ccg_sxifsm.v
 1 // pa to sx 0
                                                                                                           1 // sx to pa 0
 2 output opa to sx0 write;
                                                                                                           2 output osx_to_pa_advanceread_0;
     output [5:0] opa_to_sx0_wrdata;
                                                                                                           4 // sx_to_pa_1
5 // pa to sx 1
                                                                                                           5 output osx_to_pa_advanceread_1;
 6 output opa_to_sx1_write;
     output [5:0] opa_to_sx1_wrdata;
                                                                                                           7 // debug only
                                                                                                           8 output osx_pending_fifo_write;
 9 // position memory
                                                                                                           9 output [17:0] osx_pending_fifo_wrdata;
10 output omem_position_write;
     output [5:0] omem_position_wraddr;
                                                                                                          output [127:0] omem_position_wrdata;
                                                                                                          12 // SIGNAL DECLARATIONS
13
    output omem_point_write;
                                                                                                          15 // vgt_to_ccgen fifo
     output [5:0] omem_point_wraddr;
                                                                                                          16 reg vgt_to_ccgen_fifo_empty;
17
     output [31:0] omem_point_wrdata;
                                                                                                          17 reg [6:0] vgt_to_ccgen_active_verts;
18
                                                                                                          18 reg [2:0] vgt_to_ccgen_state_var_indx;
19 // ccg
                                                                                                          20 reg [SXIF_STATE_WIDTH-1:0] sxif_state0;
20 output odecrement available positions:
21
                                                                                                          21 reg [SXIF_STATE_WIDTH-1:0] sxif_state1;
22 // vertex fifo
                                                                                                          22 reg [SXIF_STATE_WIDTH-1:0] sxif_state2;
23 output overtex fifo write:
                                                                                                          23 reg [SXIF_STATE_WIDTH-1:0] sxif_state3;
24 output [VERTEX_FIFO_WIDTH-1:0] overtex_fifo_wrdata;
                                                                                                          24 reg [SXIF STATE WIDTH-1:0] sxif state4;
25
                                                                                                          25 reg [SXIF_STATE_WIDTH-1:0] sxif_state5;
                                       Page 7 of 48
                                                                                                                                                 Page 8 of 48
                                                                                                                                                                     Ex. 2117 - pa_ccg_sxifsm.v
                                                            Ex. 2117 - pa_ccg_sxifsm.v
```

1	reg [SXIF_STATE_WIDTH-1:0] sxif_state6;	1 // point memory
2	reg [SXIF_STATE_WIDTH-1:0] sxif_state7;	2 reg tcl_scratch_mem_point_write;
3	// vertex_fifo	3 reg [5:0] point_address;
4	reg vertex_fifo_full;	4 reg [31:0] tcl_scratch_mem_point_data;
5	reg vertex_fifo_advanceread;	5 // ccg
6	// ccg	6 reg decrement_available_positions;
7	reg [6:0] available_positions;	7 // vertex_fifo
8	// sx_to_pa	8 reg vertex_fifo_write;
9	reg [1:0] sx_to_pa_empty;	9 reg [10:0] vertex_fifo_wr_param_cache_indx;
10	reg [127:0] sx_to_pa_vector[0:1];	10 reg [2:0] vertex_fifo_wr_state_var_indx;
11		11 reg vertex_fifo_wr_edge_flag;
12	// outputs	12 reg vertex_fifo_wr_kill_flag;
13	// vgt_to_ccgen fifo	13 // sx_to_pa
14	reg vgt_to_ccgen_fifo_advanceread;	14 reg [1:0] sx_to_pa_advanceread;
15	// pa_to_sx	15 // debug only
16	reg pa_to_sx_write[0:1];	16 reg sx_pending_fifo_write;
17	reg pa_to_sx_req[0:1];	17 reg [17:0] sx_pending_fifo_wrdata;
18	reg pa_to_sx_sp_id[0:1];	18
19	reg [1:0] pa_to_sx_offset[0:1];	19 // local
20	reg pa_to_sx_aux[0:1];	20 reg [4:0] statevar_bits_vert_param_cache_size;
21	reg pa_to_sx_last[0:1];	21 reg statevar_bits_use_vtx_point_size;
22	// position memory	22 reg statevar_bits_sxpa_aux_vector;
23	reg tcl_scratch_mem_position_write;	23 reg statevar_bits_use_vtx_edge_flag;
24	reg [5:0] position_address;	24 reg statevar_bits_use_vtx_kill_flag;
25	reg [127:0] tcl_scratch_mem_position_data;	25 reg [5:0] next_sx_request_indx;
	D 0.640	D 10.640
	Page 9 of 48 Ex. 2117 - pa_ccg_sxifsm.v	Page 10 of 48 Ex. 2117 - pa_ccg_sxifsm.v
1 2 3	reg [1:0] pasx_req_ent[0:1];	1 reg [6:0] next_sx_pending_rd_pei; 2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx;
2	reg [1:0] pasx_req_cnt[0:1]; reg [1:0] next_sx_sent;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx;
2 3 4	reg [1:0] pasx_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent;	<pre>2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel;</pre>
2 3 4 5	reg [1:0] pasx_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id;
2 3 4 5 6	reg [1:0] pasx_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel;
2 3 4 5	reg [1:0] pasx_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_ine;
2 3 4 5 6 7 8	reg [1:0] pasx_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sy_id;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pei;
2 3 4 5 6	reg [1:0] pax_req_cnt[0:1]; reg [1:0] nxt_sx_sent; reg [1:0] ax_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_set;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask;
2 3 4 5 6 7 8 9	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_sel;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx;
2 3 4 4 5 6 6 7 8 9 10	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sy_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pci;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel;
2 3 4 5 6 7 8 9 10 11	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pei; reg [6:0] sx_pending_wr_pei;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_ine; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_sp_id;
2 3 4 5 6 7 8 9 10	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_su_sel; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pci; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_ine; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_sp_id; 13 reg sx_pending_rd_aux_sel;
2 3 4 5 6 7 8 9 10 11 12 13	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pci; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_sel; 14 reg sx_pending_rd_aux_inc;
2 3 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_iel; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg sx_pending_wr_pci; reg [6:0] sx_pending_wr_req_mask; reg [3:0] sx_pending_wr_red_mask; reg [2:0] sx_pending_mr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr, reg [1:0] sx_pending_fifo_rdata;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pci; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_inc; 14 reg sx_pending_rd_aux_inc; 15 reg [6:0] sx_pending_rd_pci;
2 3 4 5 6 7 8 9 10 11 12 13	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pci; reg [6:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_rdaddr; reg [1:0] sx_pending_fifo_rdaddr;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pci; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_inc; 14 reg sx_pending_rd_aux_inc; 15 reg [6:0] sx_pending_rd_pci; 16 reg [3:0] sx_pending_rd_req_mask;
2 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] sx_sent; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_pei; reg [6:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fio_rdaddr; reg [1:0] sx_pending_fifo_rdaddr; reg [1:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_contents;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_ine; 8 reg [6:0] initial_sx_pending_rd_pei; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_ine; 14 reg sx_pending_rd_aux_ine; 15 reg [6:0] sx_pending_rd_pei; 16 reg [3:0] sx_pending_rd_req_mask; 17 reg [2:0] sx_pending_rd_state_var_indx;
2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] sx_sent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_ret_indx; reg [3:0] sx_pending_wr_ret_indx; reg [1:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_rddata; reg [1:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_ontents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_ontents; reg [1:0] sx_pending_fifo_ontents;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pci; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_inc; 14 reg sx_pending_rd_pci; 15 reg [6:0] sx_pending_rd_pci; 16 reg [3:0] sx_pending_rd_req_mask; 17 reg [2:0] sx_pending_rd_state_var_indx; 18 reg next_aux_sel;
2 3 3 4 5 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_req_mask; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_rddata; reg [1:0] sx_pending_fifo_rddata; reg [1:0] sx_pending_fifo_rdaddr; reg [2:0] sx_pending_fifo_ontents; reg [1:0] sx_pending_fifo_ontents; reg [1:0] sx_pending_fifo_empty;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pci; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_inc; 15 reg [6:0] sx_pending_rd_pci; 16 reg [3:0] sx_pending_rd_req_mask; 17 reg [2:0] sx_pending_rd_state_var_indx; 18 reg next_aux_sel; 19 reg aux_sel;
2 3 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_red_mask; reg [1:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_todata; reg [1:0] sx_pending_fifo_todata; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg sx_pending_fifo_empty; reg sx_pending_fifo_fo_fill;	2 reg [3:0] next_sx_pending_rd_req_mask; 3 reg [2:0] next_sx_pending_rd_state_var_indx; 4 reg initial_sx_pending_rd_sx_sel; 5 reg initial_sx_pending_rd_sp_id; 6 reg initial_sx_pending_rd_aux_sel; 7 reg initial_sx_pending_rd_aux_inc; 8 reg [6:0] initial_sx_pending_rd_pci; 9 reg [3:0] initial_sx_pending_rd_req_mask; 10 reg [2:0] initial_sx_pending_rd_state_var_indx; 11 reg sx_pending_rd_sx_sel; 12 reg sx_pending_rd_aux_sel; 13 reg sx_pending_rd_aux_inc; 15 reg [6:0] sx_pending_rd_pci; 16 reg [3:0] sx_pending_rd_req_mask; 17 reg [2:0] sx_pending_rd_state_var_indx; 18 reg next_aux_sel; 19 reg aux_sel; 20 reg [6:0] next_param_cache_base;
2 3 3 4 4 5 6 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pci; reg [6:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_rdaddr; reg [1:0] sx_pending_fifo_rdaddr; reg [1:0] sx_pending_fifo_tdaddr; reg [1:0] sx_pending_fifo_fontents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg sx_pending_fifo_advanceread;	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_sp_id; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_inc; reg [6:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_state_var_indx; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_inc; reg sx_pending_rd_pending_rd_state_var_indx; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_inc; reg [6:0] sx_pending_rd_pending_rd_state_var_indx; reg [2:0] sx_pending_rd_state_var_indx; reg [2:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] next_param_cache_base; reg [6:0] param_cache_base;
2 3 3 4 4 5 5 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_ncpi; reg [6:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_fidata; reg [1:0] sx_pending_fifo_todata; reg [1:0] sx_pending_fifo_todata; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_todata; reg sx_pending_fifo_todata; reg sx_pending_fifo_todata; reg sx_pending_fifo_todata; reg sx_pending_fifo_contents; reg sx_pending_fifo_todata; reg sx_pending_fifo_todata; reg sx_pending_fifo_todata; reg sx_pending_fifo_todata; reg sx_pending_fifo_datanceread; reg next_sx_pending_fifo_datanceread; reg next_sx_pending_fifo_dats_sel;	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_sp_id; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_ine; reg [6:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_state_var_indx; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg sx_pending_rd_pending_rd_state_var_indx; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg [6:0] sx_pending_rd_pending_rd_state_var_indx; reg [2:0] sx_pending_rd_state_var_indx; reg [2:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] next_param_cache_base; reg [6:0] param_cache_base; reg next_sx_aux;
2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_preq_mask; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_findata; reg [1:0] sx_pending_fifo_todata; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_daddr; reg sx_pending_fifo_dadvanceread; reg sx_pending_fifo_advanceread; reg next_sx_pending_rd_sx_sel; reg next_sx_pending_rd_sp_id;	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_sp_id; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_ine; reg [6:0] initial_sx_pending_rd_req_mask; reg [3:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_state_var_indx; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_req_mask; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg next_sx_aux; reg sx_aux;
2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_peri; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_fidata; reg [1:0] sx_pending_fifo_tanddr; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_daddr; reg sx_pending_fifo_daddaddr; reg sx_pending_fifo_daddaddr; reg sx_pending_fifo_daddaddr; reg sx_pending_fifo_daddadddddddddddddddddddddddddddddddd	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_sp_id; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_ine; reg [6:0] initial_sx_pending_rd_pei; reg [3:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_state_var_indx; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_req_mask; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg sx_aux; reg [2:0] next_sx_receive_indx;
2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	reg [1:0] pax_req_ent[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_ent; reg [1:0] decrement_pasx_req_ent; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sp_id; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_preq_mask; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_findata; reg [1:0] sx_pending_fifo_todata; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_daddr; reg sx_pending_fifo_dadvanceread; reg sx_pending_fifo_advanceread; reg next_sx_pending_rd_sx_sel; reg next_sx_pending_rd_sp_id;	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_sp_id; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_ine; reg [6:0] initial_sx_pending_rd_req_mask; reg [3:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_state_var_indx; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_req_mask; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg [6:0] param_cache_base; reg next_sx_aux; reg sx_aux;
2 3 3 4 4 5 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	reg [1:0] pax_req_cnt[0:1]; reg [1:0] next_sx_sent; reg [1:0] increment_pasx_req_cnt; reg [1:0] decrement_pasx_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_aux_ine; reg sx_pending_wr_aux_ine; reg [6:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_textate_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_taddr; reg [1:0] sx_pending_fifo_taddr; reg [1:0] sx_pending_fifo_taddr; reg [2:0] sx_pending_fifo_taddr; reg next_sx_pending_rd_sx_sel; reg next_sx_pending_rd_aux_sel; reg next_sx_pending_rd_aux_ine; Page 11 of 48	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_sp_id; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_ine; reg [6:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_req_mask; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg sx_pending_rd_pending_rd_pending_reg [3:0] sx_pending_rd_pending_rd_state_var_indx; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_ine; reg [3:0] sx_pending_rd_pending_rd_req_mask; reg [3:0] sx_pending_rd_req_mask; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] sx_pending_rd_state_var_indx; reg [6:0] next_param_cache_base; reg [6:0] next_param_cache_base; reg [6:0] param_cache_base; reg [6:0] param_cache_base;
2 3 3 4 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	reg [1:0] pax_req_cnt[0:1]; reg [1:0] nx_sent; reg [1:0] increment_pax_req_cnt; reg [1:0] decrement_pax_req_cnt; reg sx_pending_wr_sx_sel; reg sx_pending_wr_sx_sel; reg sx_pending_wr_aux_sel; reg sx_pending_wr_aux_inc; reg [6:0] sx_pending_wr_pci; reg [3:0] sx_pending_wr_req_mask; reg [2:0] sx_pending_wr_state_var_indx; reg [1:0] sx_pending_fifo_wraddr; reg [1:0] sx_pending_fifo_taddat; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_contents; reg [1:0] sx_pending_fifo_daddr; reg sx_pending_fifo_daddr; reg sx_pending_fifo_daddr; reg sx_pending_fifo_fo_dadd; reg sx_pending_fifo_fo_fo_daddr; reg sx_pending_fifo_fo_fo_fo_fo_fo_fo_fo_fo_fo_fo_fo_fo_f	reg [3:0] next_sx_pending_rd_req_mask; reg [2:0] next_sx_pending_rd_state_var_indx; reg initial_sx_pending_rd_sx_sel; reg initial_sx_pending_rd_aux_sel; reg initial_sx_pending_rd_aux_inc; reg [6:0] initial_sx_pending_rd_pei; reg [3:0] initial_sx_pending_rd_req_mask; reg [2:0] initial_sx_pending_rd_state_var_indx; reg sx_pending_rd_sx_sel; reg sx_pending_rd_aux_sel; reg sx_pending_rd_aux_inc; reg sx_pending_rd_aux_inc; reg [6:0] sx_pending_rd_pei; reg [6:0] sx_pending_rd_state_var_indx; reg sx_aux; reg sx_aux; reg sx_aux; reg sx_aux; reg sx_aux; reg [2:0] sx_receive_indx;

1	reg next_sx_pending_advance;	1 reg [2:0] var300_sx_pending_fifo_contents;
2	reg sx_pending_advance;	2 reg [2:0] var400_next_sx_receive_indx;
3	reg [5:0] next_point_address;	3 reg var400_ignore_this_cycle;
4	reg [5:0] next_position_address;	4 reg var400_sx_sel;
5	reg [3:0] vertex_fifo_entriesavailable;	5 reg var400_vector_write;
6	reg next_pa_to_sx_write[0:1];	6 reg var400_first_vector_write;
7	reg next_pa_to_sx_req[0:1];	7 reg var400_last_vector_write;
8	reg next_pa_to_sx_sp_id[0:1];	8 reg [2:0] var400_vector_wr_cnt;
9	reg [1:0] next_pa_to_sx_offset[0:1];	9 reg var400_next_sx_pending_rd_sx_sel;
10	reg next_pa_to_sx_aux[0:1];	10 reg var400_next_sx_pending_rd_sp_id;
11	reg next_pa_to_sx_last[0:1];	11 reg var400_next_sx_pending_rd_aux_sel;
12	reg [2:0] request_side_vs_export_mode;	12 reg var400_next_sx_pending_rd_aux_inc;
13	reg [3:0] request_side_vs_export_count;	13 reg [6:0] var400_next_sx_pending_rd_pci;
14	reg [2:0] receive_side_vs_export_mode;	14 reg [3:0] var400_next_sx_pending_rd_req_mask;
15		15 reg [2:0] var400_next_sx_pending_rd_state_var_indx;
16	// local variables	16 reg [1:0] var480_next_pasx_req_cnt_0;
17	reg [SXIF_STATE_WIDTH-1:0] var80_sxif_state;	17 reg [1:0] var480_next_pasx_req_cnt_1;
18	reg [SXIF_STATE_WIDTH-1:0] var85_sxif_state;	18 reg [3:0] var600_vertex_fifo_entriesavailable;
19	reg [5:0] var00_vgt_to_ccgen_active_verts;	19 reg [127:0] var400_sx_to_pa_vector_var400_sx_sel;
20	reg var100_sx_sel;	20 reg [31:0] var400_sx_to_pa_vector_y_var400_sx_sel;
21	reg var100_sp_id;	21 reg [31:0] var400_sx_to_pa_vector_z_var400_sx_sel;
22	reg [1:0] var100_offset;	22
23	reg [6:0] var100_pci;	23 // for-loop variables
24	reg [7:0] var100_remaining_positions;	24 integer indx100;
25	reg var100_sx_all_sent;	25 integer indx400;
	Page 13 of 48	Page 14 of 48
	Ex. 2117 - pa_ccg_sxifsm.v	Ex. 2117 - pa_ccg_sxifsm.v
1	integer indx477;	
1 2	integer indx477;	
	integer indx477;	2 // vgt_to_ccgen fifo
2		 2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty;
2		 2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty;
2 3 4	//////////////////////////////////////	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts,
2 3 4 5	//////////////////////////////////////	<pre>2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {l'b0, var00_vgt_to_ccgen_active_verts};</pre>
2 3 4 5 6	//////////////////////////////////////	<pre>2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1b0, var00_vgt_to_ccgen_active_verts};</pre>
2 3 4 5 6 7	///map inputs // map ways @(ivgt_to_ccgen_fifo_empty or	<pre>2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1¹b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin</pre>
2 3 4 5 6 7 8	// map inputs // map inputs // map inputs // ivgt_to_ccgen_fifo_empty or // ivgt_to_ccgen_fifo_rddata or	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1100, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT;
2 3 4 5 6 7 8	// map inputs // map inputs // ivgt_to_ccgen_fifo_empty or ivgt_to_ccgen_fifo_rddata or isxif_state0 or	<pre>2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end</pre>
2 3 4 5 6 7 8 9	// map inputs // map inputs // ivgt_to_ccgen_fifo_empty or ivgt_to_ccgen_fifo_rddata or isxif_state0 or isxif_state1 or	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10
2 3 4 5 6 7 8 9 10	// map inputs // ivgt_to_ccgen_fifo_empty or ivgt_to_ccgen_fifo_empty or ivgt_to_ccgen_fifo_rddata or isxif_state0 or isxif_state1 or isxif_state2 or	<pre>2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state</pre>
2 3 4 5 6 7 8 9 10 11	// map inputs // map inputs always @(ivgt_to_ccgen_fifo_empty or ivgt_to_ccgen_fifo_rddata or isxif_state0 or isxif_state1 or isxif_state2 or isxif_state3 or	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0;
2 3 4 5 6 7 8 9 10 11 12 13	// map inputs // map	<pre>2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1;</pre>
2 3 4 5 6 7 8 9 10 11 12 13	// map inputs // input i	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2;
2 3 4 5 6 7 8 9 10 11 12 13 14	// map inputs // ivgt_to_ccgen_fifo_empty or // ivgt_to_ccge	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx) = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state3;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	// map inputs // map	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state3; 16 sxif_state4 = isxif_state4;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	// map inputs // map	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	// map inputs // map	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {11b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 1h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5; 18 sxif_state6 = isxif_state6;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	// map inputs // with a construction of the constr	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5; 18 sxif_state6 = isxif_state6; 19 sxif_state7 = ixxif_state7;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	// map inputs //	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx) = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {1/50, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5; 18 sxif_state6 = isxif_state6; 19 sxif_state7 = isxif_state7; 20 21 // vertex_fifo
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	// map inputs //	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {I'b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state5; 18 sxif_state5 = isxif_state5; 19 sxif_state6 = isxif_state7; 20 21 // vertex_fifo_full = ivertex_fifo_full;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	// map inputs // map	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx) = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {17b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 7b0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5; 18 sxif_state6 = isxif_state6; 19 sxif_state7 = isxif_state7; 20 21 // vertex_fifo_full = ivertex_fifo_full;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	// map inputs //	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {11b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 1h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5; 18 sxif_state5 = isxif_state6; 19 sxif_state7 = isxif_state7; 20 21 // vertex_fifo 22 vertex_fifo_full = ivertex_fifo_fall; 23 vertex_fifo_advanceread = ivertex_fifo_advanceread;
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	// map inputs // map input substituted // map inputs // map inp	2 // vgt_to_ccgen fifo 3 vgt_to_ccgen_fifo_empty = ivgt_to_ccgen_fifo_empty; 4 {var00_vgt_to_ccgen_active_verts, 5 vgt_to_ccgen_state_var_indx} = ivgt_to_ccgen_fifo_rddata; 6 vgt_to_ccgen_active_verts = {11b0, var00_vgt_to_ccgen_active_verts}; 7 if (var00_vgt_to_ccgen_active_verts == 'h0) begin 8 vgt_to_ccgen_active_verts = VERTICES_PER_SLOT; 9 end 10 11 // state 12 sxif_state0 = isxif_state0; 13 sxif_state1 = isxif_state1; 14 sxif_state2 = isxif_state2; 15 sxif_state3 = isxif_state2; 16 sxif_state4 = isxif_state4; 17 sxif_state5 = isxif_state5; 18 sxif_state6 = isxif_state6; 19 sxif_state7 = isxif_state7; 20 21 // vertex_fifo 22 vertex_fifo_full = ivertex_fifo_full; 23 vertex_fifo_advanceread = ivertex_fifo_advanceread; 24

```
1
      available positions = iavailable positions;
2
                                                                                                              2 // pa_to_sx 1
3
     // sx to pa 0
                                                                                                              3 assign opa_to_sx1_write = pa_to_sx_write[1];
4
     sx_to_pa_empty[0] = isx_to_pa_empty_0;
                                                                                                              4 assign opa_to_sx1_wrdata = {pa_to_sx_req[1],
5
      sx_to_pa_vector[0] = isx_to_pa_vector_0;
                                                                                                                                pa_to_sx_sp_id[1],
                                                                                                                                pa_to_sx_offset[1],
                                                                                                                                pa_to_sx_aux[1],
      sx_to_pa_empty[1] = isx_to_pa_empty_1;
                                                                                                                                pa_to_sx_last[1]};
      sx_to_pa_vector[1] = isx_to_pa_vector_1;
11
                                                                                                             11 assign omem_position_write = tcl_scratch_mem_position_write;
12
                                                                                                             12 assign omem_position_wraddr = position_address;
13
     assign omem_position_wrdata = tcl_scratch_mem_position_data;
     15
                                                                                                             15 // point memory
     // vgt to ccgen fifo
                                                                                                             16 assign omem point write = tcl scratch mem point write;
16
17
     assign\ ovgt\_to\_ccgen\_fifo\_advanceread = vgt\_to\_ccgen\_fifo\_advanceread;
                                                                                                             17 assign omem_point_wraddr = point_address;
18
                                                                                                             18
                                                                                                                 assign omem point wrdata = tcl scratch mem point data;
19
    // pa to sx 0
                                                                                                             19
                                                                                                             20 // ccg
20
     assign opa_to_sx0_write = pa_to_sx_write[0];
21
     assign\ opa\_to\_sx0\_wrdata = \{pa\_to\_sx\_req[0],
                                                                                                             21 assign odecrement_available_positions = decrement_available_positions;
22
                  pa_to_sx_sp_id[0],
                                                                                                             22
23
                  pa_to_sx_offset[0],
                                                                                                             23 // vertex fifo
24
                  pa_to_sx_aux[0],
                                                                                                             24
                                                                                                                  assign overtex_fifo_write = vertex_fifo_write;
                  pa_to_sx_last[0]};
                                                                                                                  assign overtex_fifo_wrdata = {vertex_fifo_wr_param_cache_indx,
                                        Page 17 of 48
                                                                                                                                                     Page 18 of 48
                                                              Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                           Ex. 2117 - pa_ccg_sxifsm.v
                    vertex_fifo_wr_state_var_indx,
                                                                                                                      ) begin : proc80
 2
                    vertex fifo wr edge flag,
                                                                                                                  case(vgt_to_ccgen_state_var_indx)
                    vertex fifo wr kill flag};
                                                                                                                    0 : var80 sxif state = sxif state0
                                                                                                                    1 : var80 sxif state = sxif state1:
                                                                                                                    2 : var80 sxif state = sxif state2
 6 assign osx_to_pa_advanceread_0 = sx_to_pa_advanceread[0];
                                                                                                                    3 : var80 sxif state = sxif state3;
     assign osx_to_pa_advanceread_1 = sx_to_pa_advanceread[1];
                                                                                                                    4 : var80 sxif state = sxif state4
                                                                                                                    5 : var80 sxif state = sxif state5;
    // debug only
                                                                                                                    6 : var80_sxif_state = sxif_state6;
     assign osx_pending_fifo_write = sx_pending_fifo_write;
10
                                                                                                             10
                                                                                                                    7 : var80_sxif_state = sxif_state7;
11
     assign osx_pending_fifo_wrdata = sx_pending_fifo_wrdata;
                                                                                                             11
                                                                                                                    default : var80_sxif_state = sxif_state0;
12
                                                                                                             12
     13
14 // state mux, request side
                                                                                                                   request_side_vs_export_mode = var80_sxif_state[6:4];
     request_side_vs_export_count = var80_sxif_state[3:0];
17
         sxif_state0 or
18
         sxif_state1 or
                                                                                                                 19
         sxif_state2 or
                                                                                                                 // state mux, receive side
                                                                                                             20
         sxif state3 or
21
         sxif_state4 or
                                                                                                             21 always @(
22
         sxif state5 or
                                                                                                             22
                                                                                                                      sxif state0 or
23
         sxif state6 or
                                                                                                             23
                                                                                                                      sxif state1 or
24
         sxif state7 or
                                                                                                             24
                                                                                                                      sxif state2 or
25
         vgt_to_ccgen_state_var_indx
                                                                                                                       sxif state3 or
                                        Page 19 of 48
                                                                                                                                                     Page 20 of 48
                                                              Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                           Ex. 2117 - pa_ccg_sxifsm.v
```

```
sxif state4 or
                                                                                                                    request_side_vs_export_mode or
 2
          sxif state5 or
                                                                                                                    request_side_vs_export_count
          sxif state6 or
                                                                                                                   ) begin : proc90
          sxif state7 or
          next_sx_pending_rd_state_var_indx
                                                                                                           5 statevar_bits_vert_param_cache_size = request_side_vs_export_count + 1;
         ) begin : proc85
      case(next_sx_pending_rd_state_var_indx)
                                                                                                           7 if ((request_side_vs_export_mode != 0) &&
      0 : var85_sxif_state = sxif_state0;
                                                                                                                 (request_side_vs_export_mode != 7)) begin
      1 : var85_sxif_state = sxif_state1;
                                                                                                                statevar_bits_sxpa_aux_vector = 'h1;
      2 : var85_sxif_state = sxif_state2;
                                                                                                          10
11
     3 : var85_sxif_state = sxif_state3;
12 4 : var85_sxif_state = sxif_state4;
                                                                                                                statevar_bits_sxpa_aux_vector = 'h0;
13
     5 : var85_sxif_state = sxif_state5;
      6 : var85_sxif_state = sxif_state6;
15
      7 : var85_sxif_state = sxif_state7;
                                                                                                          15
      default : var85_sxif_state = sxif_state0;
                                                                                                          16
17
                                                                                                          17 // decode receive side state
      endcase
18
                                                                                                          19
      receive_side_vs_export_mode = var85_sxif_state[6:4];
                                                                                                          19 always @(
20 end
                                                                                                          20
                                                                                                                    receive_side_vs_export_mode
21
                                                                                                          21
                                                                                                                  ) begin : proc95
22
    22 if ((receive side vs export mode == 2) ||
23 // decode request side state
                                                                                                          23
                                                                                                                 (receive_side_vs_export_mode == 5)) begin
24
                                                                                                                statevar_bits_use_vtx_point_size = 'h1;
25 always @(
                                                                                                          25
                                       Page 21 of 48
                                                                                                                                                 Page 22 of 48
                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                       Ex. 2117 - pa_ccg_sxifsm.v
     else begin
                                                                                                            1 always @(
 2
       statevar_bits_use_vtx_point_size = 'h0;
                                                                                                                    aux sel or
 3
                                                                                                                    next_sx_request_indx or
4
                                                                                                                    pa_to_sx_last[0] or
 5 \qquad \text{if ((receive\_side\_vs\_export\_mode == 3) } \parallel
                                                                                                                    pa_to_sx_last[1] or
        (receive_side_vs_export_mode == 6)) begin
                                                                                                                    param_cache_base or
7
       statevar\_bits\_use\_vtx\_edge\_flag = \text{'}h1;
                                                                                                                    pasx_req_cnt[0] or
8
                                                                                                                    pasx_req_cnt[1] or
      else begin
                                                                                                                    statevar_bits_sxpa_aux_vector or
10
       statevar_bits_use_vtx_edge_flag = 'h0;
                                                                                                           10
                                                                                                                    statevar_bits_vert_param_cache_size or
11
                                                                                                          11
12
                                                                                                          12
                                                                                                                    sx_request_indx or
13 if ((receive_side_vs_export_mode == 4) ||
14
       (receive_side_vs_export_mode == 5) ||
                                                                                                                    var100_offset or
15
        (receive_side_vs_export_mode == 6)) begin
                                                                                                                    var100_pci or
16
                                                                                                                    var100_remaining_positions or
      statevar_bits_use_vtx_kill_flag = 'h1;
17
                                                                                                                    var100_sp_id or
18
                                                                                                                    var100_sx_sel or
19
      statevar_bits_use_vtx_kill_flag = 'h0;
                                                                                                                    vgt to ccgen active verts or
20
                                                                                                          20
      end
                                                                                                                   vgt to ccgen fifo empty or
21
     end
                                                                                                                   vgt to ccgen state var indx
22
                                                                                                                  ) begin : proc100
23
    23 // init variables
24 // PaSxRequest
                                                                                                          24 var100 sx sel = 'h0:
    var100 sp id = 'h0;
                                       Page 23 of 48
                                                                                                                                                 Page 24 of 48
                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                       Ex. 2117 - pa_ccg_sxifsm.v
```

```
1 var100 offset = 'h0;
                                                                                                                 1 sx_pending_fifo_write = 'h0;
 2
      var100\_pci = param\_cache\_base + (var100\_offset * statevar\_bits\_vert\_param\_cache\_size);
                                                                                                                 2 sx_pending_wr_sx_sel = 'h0;
       var100_remaining_positions = 'h0;
                                                                                                                 3 sx_pending_wr_sp_id = 'h0;
       var100_sx_all_sent = 'h1;
                                                                                                                 4 sx_pending_wr_aux_sel = 'h0;
                                                                                                                 5 sx_pending_wr_aux_inc = 'h0;
 6 // defaults
                                                                                                                 6 sx_pending_wr_pci = 'h0;
                                                                                                                 7 sx_pending_wr_req_mask = 'hf;
 7 increment_pasx_req_cnt[0] = 'h0;
                                                                                                                 8 sx_pending_wr_state_var_indx = vgt_to_ccgen_state_var_indx;
 8 increment_pasx_req_cnt[1] = 'h0;
 9 next_pa_to_sx_req[0] = 'h0;
                                                                                                                 9     next_sx_sent[0] = sx_sent[0];
10     next_sx_sent[1] = sx_sent[1];
11     next_pa_to_sx_offset[0] = 'h0;
12     next_pa_to_sx_aux[0] = 'h0;
                                                                                                                12 if (vgt_to_ccgen_fifo_empty == 'h0) begin
var100_sx_sel = sx_request_indx[SX_SEL_BIT];
var100_sp_id = sx_request_indx[SP_ID_BIT];
15 next pa to sx reg[1] = 'h0;
                                                                                                                15
                                                                                                                      var100 offset = sx request indx[OFFSET HI BIT:OFFSET LO BIT];
16 next pa to sx sp id[1] = 'h0;
17 next pa to sx offset[1] = 'h0:
                                                                                                                17
                                                                                                                       var100 pci = param cache base + (var100 offset * statevar bits vert param cache size);
18 next_pa_to_sx_aux[1] = 'h0;
                                                                                                                18
19     next_pa_to_sx_last[1] = 'h0;
                                                                                                                19
                                                                                                                       if (pasx req cnt[var100 sx sel] < ACTIVE PASX REQUESTS) begin
20 next pa to sx write[1] = 'h0;
                                                                                                                20
                                                                                                                        increment_pasx_req_cnt[var100_sx_sel] = 'h1;
21 next aux sel = aux sel;
                                                                                                               21
22 vgt to ccgen fifo advanceread = 'h0;
                                                                                                               22
                                                                                                                        var100_remaining_positions = vgt_to_ccgen_active_verts - sx_request_indx;
23 next_sx_request_indx = sx_request_indx;
                                                                                                               23
                                                                                                                        if \, (var100\_remaining\_positions[7] == \text{'h1}) \, begin
24
      next_param_cache_base = param_cache_base;
                                                                                                               24
                                                                                                                        var100_remaining_positions = 'h0;
       next_sx_aux = sx_aux;
                                                                                                                25
                                         Page 25 of 48
                                                                                                                                                        Page 26 of 48
                                                                                                                                                                               Ex. 2117 - pa_ccg_sxifsm.v
                                                               Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                         3 : begin
 2
        sx pending wr sx sel = var100 sx sel;
                                                                                                                 2
                                                                                                                          sx_pending_wr_req_mask = 'h7;
        sx_pending_wr_sp_id = var100_sp_id;
                                                                                                                 3
        sx\_pending\_wr\_aux\_sel = aux\_sel;
                                                                                                                         default : begin
        sx_pending_wr_aux_inc = 'h0;
                                                                                                                          sx_pending_wr_req_mask = 'hf;
                                                                                                                 7
        if (statevar_bits_sxpa_aux_vector == 'h0) begin
 8
         sx_pending_wr_aux_inc = 'h1;
                                                                                                                 8
                                                                                                                        endcase
10
                                                                                                                10
                                                                                                                        sx_pending_wr_state_var_indx = vgt_to_ccgen_state_var_indx;
11
         sx_pending_wr_pci = var100_pci;
                                                                                                               11
12
                                                                                                               12
                                                                                                                        sx_pending_fifo_write = 'h1;
13
         case(var100_remaining_positions)
                                                                                                                13
14
                                                                                                                14
                                                                                                                        next_pa_to_sx_req[var100_sx_sel] = 'h1;
15
          sx_pending_wr_req_mask = 'h0;
                                                                                                                        next_pa_to_sx_sp_id[var100_sx_sel] = var100_sp_id;
16
                                                                                                                        next_pa_to_sx_offset[var100_sx_sel] = var100_offset;
17
                                                                                                                        next_pa_to_sx_aux[var100_sx_sel] = sx_aux;
18
19
          sx_pending_wr_req_mask = 'h1;
                                                                                                                        next_pa_to_sx_last[var100_sx_sel] = 'h0;
                                                                                                                        if (var100_remaining_positions <= VECTORS_PER_SX_REQUEST_SET) begin
20
                                                                                                                20
21
                                                                                                                21
                                                                                                                        next pa to sx last[var100 sx sel] = 'h1;
22
                                                                                                                22
23
                                                                                                                23
          sx_pending_wr_req_mask = 'h3;
24
                                                                                                                24
          end
                                                                                                                        next_pa_to_sx_write[var100_sx_sel] = 'h1;
25
                                         Page 27 of 48
                                                                                                                                                        Page 28 of 48
                                                               Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                               Ex. 2117 - pa_ccg_sxifsm.v
```

```
var100_sx_all_sent = 'h1;
                                                                                                                       end
2
        for (indx100 = 0: indx100 < PASX_INTERFACES: indx100 = indx100 + 1) begin
                                                                                                               2
                                                                                                                       end
     else begin
                                                                                                                       next_aux_sel = 'h1;
        if \, ((aux\_sel == 'h1)
                                                                                                                      if (next_pa_to_sx_last[var100_sx_sel] == 'h1) begin
          (statevar_bits_sxpa_aux_vector == 'h0)) begin
         next_aux_sel = 'h0;
10
                                                                                                              10
                                                                                                                      else\ if\ (statevar\_bits\_sxpa\_aux\_vector == 'h1)\ begin
     if ((var100_remaining_positions <= VECTORS_PER_SX_REQUEST) && (var100_sx_all_sent == 'h1)) begin
                                                                                                                       next_sx_aux = ~sx_aux;
12
13
          vgt to ccgen fifo advanceread = 'h1;
14
15
          next sx request indx = 'h0;
                                                                                                              15 end
16
                                                                                                              16
17
          next_param_cache_base = param_cache_base +
                                                                                                              18
                      (statevar\_bits\_vert\_param\_cache\_size*US\_ALU\_VE\_MEMORIES);
                                                                                                              18 // sx_pending_fifo_wrdata
19
20
          for (indx100 = 0; indx100 < PASX_INTERFACES; indx100 = indx100 + 1) begin
                                                                                                              21
           next_sx_sent[indx100] = 'h0;
                                                                                                              20 always @(
22
                                                                                                             21
                                                                                                                        sx_pending_wr_sx_sel or
23
          end
                                                                                                             22
                                                                                                                        sx_pending_wr_sp_id or
24
                                                                                                             23
                                                                                                                       sx_pending_wr_aux_sel or
25
          next_sx_request_indx = next_sx_request_indx + VECTORS_PER_SX_REQUEST;
                                                                                                             24
                                                                                                                       sx_pending_wr_aux_inc or
                                                                                                              25
                                                                                                                        sx_pending_wr_pci or
          next_sx_sent[var100_sx_sel] = 'h1;
                                        Page 29 of 48
                                                                                                                                                      Page 30 of 48
                                                              Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                     var300_sx_pending_fifo_contents = sx_pending_fifo_contents;
          sx pending wr req mask or
 2
                                                                                                               2
          sx _pending_wr_state_var_indx
                                                                                                                     if (sx pending fifo write == 'h1) begin
         ) begin: proc200
      sx pending fifo wrdata = {
                                                                                                                      sx\_pending\_fifo[sx\_pending\_fifo\_wraddr] \mathrel{<=} sx\_pending\_fifo\_wrdata;
                  sx_pending_wr_sx_sel,
                                                                                                                      sx\_pending\_fifo\_wraddr \mathrel{<=} sx\_pending\_fifo\_wraddr + \text{'}h1;
                  sx_pending_wr_sp_id,
                                                                                                                      var300_sx_pending_fifo_contents = var300_sx_pending_fifo_contents + 'h1;
                                                                                                              7
                  sx_pending_wr_aux_sel,
                  sx_pending_wr_aux_inc,
                                                                                                               8
                  sx_pending_wr_pci,
                                                                                                                     if (sx\_pending\_fifo\_advanceread == 'h1) \ begin
10
                  sx_pending_wr_req_mask,
                                                                                                              10
                                                                                                                      sx\_pending\_fifo\_rdaddr <= sx\_pending\_fifo\_rdaddr + 'h1;
11
                  sx_pending_wr_state_var_indx
                                                                                                             11
                                                                                                                      var300\_sx\_pending\_fifo\_contents = var300\_sx\_pending\_fifo\_contents - 'h1;
12
                                                                                                              12
13 end
                                                                                                              13
14
                                                                                                              14
                                                                                                                     if \left( (var300\_sx\_pending\_fifo\_contents == (SX\_PENDING\_FIFO\_DEPTH-1) \right) \parallel
15
                                                                                                                       (var300\_sx\_pending\_fifo\_contents == SX\_PENDING\_FIFO\_DEPTH)) \qquad begin
     sx\_pending\_fifo\_full \mathrel{<=} 'h1;
17
    // sx_pending fifo
                                                                                                              17
18
                                                                                                                     else begin
19
    always @(posedge clk) begin : proc300
                                                                                                                      sx_pending_fifo_full <= 'h0;
     if (reset == 'h1) begin
20
                                                                                                              20
21
      sx pending fifo wraddr <= 'h0;
                                                                                                             21
22
                                                                                                              22
      sx pending fifo rdaddr <= 'h0;
                                                                                                                     if (var300 sx pending fifo contents == 'h0) begin
       sx_pending_fifo_contents <= 'h0;
23
                                                                                                             23
                                                                                                                     sx pending fifo empty <= 'h1;
24
                                                                                                              24
      end
                                                                                                                      end
25
      else begin
                                                                                                              25
                                                                                                                     else begin
                                        Page 31 of 48
                                                                                                                                                      Page 32 of 48
                                                                                                                                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                              Ex. 2117 - pa_ccg_sxifsm.v
```

```
1
        sx_pending_fifo_empty <= 'h0;
                                                                                                            1 // sx_pending_wr_aux_inc
2
                                                                                                            2 // sx_pending_wr_pci
3
                                                                                                            3 // sx_pending_wr_req_mask
4
       sx_pending_fifo_contents <= var300_sx_pending_fifo_contents;
                                                                                                            4 // sx_pending_wr_state_var_indx
 5
                                                                                                            sx_pending_fifo_rddata
                                                                                                                    ) begin : proc360
                                                                                                            9 {initial_sx_pending_rd_sx_sel,
    // sx_pending fifo read mux
                                                                                                           10 initial_sx_pending_rd_sp_id,
11
     11 initial_sx_pending_rd_aux_sel,
12
    always @(
                                                                                                           12 initial_sx_pending_rd_aux_inc,
13
         sx_pending_fifo[0] or
                                                                                                           13 initial_sx_pending_rd_pci,
14
         sx_pending_fifo[1] or
                                                                                                                initial_sx_pending_rd_req_mask,
15
                                                                                                           15
         sx pending fifo[2] or
                                                                                                                initial_sx_pending_rd_state_var_indx} = sx_pending_fifo_rddata;
         sx_pending_fifo[3] or
                                                                                                           16 end
16
17
         sx pending fifo rdaddr
                                                                                                           17
                                                                                                           18
         ) begin: proc340
19
      sx_pending_fifo_rddata = sx_pending_fifo[sx_pending_fifo_rdaddr];
                                                                                                           19 // PaSxReceive
20 end
                                                                                                           21
                                                                                                           21 always @(
22
                                                                                                                     initial_sx_pending_rd_sx_sel or
23 // sx_pending_wr_sx_sel
                                                                                                           23
                                                                                                                    initial_sx_pending_rd_sp_id or
24 // sx_pending_wr_sp_id
                                                                                                           24
                                                                                                                    initial_sx_pending_rd_aux_sel or
25 // sx_pending_wr_aux_sel
                                                                                                                     initial_sx_pending_rd_aux_inc or
                                       Page 33 of 48
                                                                                                                                                  Page 34 of 48
                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                        Ex. 2117 - pa_ccg_sxifsm.v
          initial_sx_pending_rd_pci or
                                                                                                                     tel scratch mem position data or
 2
          initial_sx_pending_rd_req_mask or
                                                                                                                     available positions or
                                                                                                                     var400 first_vector_write or
          initial_sx_pending_rd_state_var_indx or
          next_point_address or
                                                                                                                     var400 last_vector_write or
          next_sx_pending_advance or
                                                                                                                     var400 sx sel or
                                                                                                                     var400_vector_wr_cnt or
          next_sx_pending_rd_req_mask or
          next_sx_receive_indx or
                                                                                                                     var400 vector write or
          point address or
                                                                                                                     vertex fifo entriesavailable or
          position_address or
                                                                                                                     vertex_fifo_full
10
          statevar_bits_use_vtx_point_size or
                                                                                                           10
                                                                                                                    ) begin: proc400
11
          statevar_bits_use_vtx_edge_flag or
                                                                                                           11
12
          statevar_bits_use_vtx_kill_flag or
                                                                                                           12 // init variables
          sx_pending_advance or
13
14
          sx_pending_fifo_empty or
                                                                                                           14 var400_vector_write = 'h0;
15
                                                                                                           var400_first_vector_write = 'h0;
          sx_pending_rd_aux_inc or
16
          sx_pending_rd_aux_sel or
                                                                                                           16 var400_last_vector_write = 'h0;
17
          sx_pending_rd_pci or
                                                                                                           17 var400_vector_wr_cnt = 'h0;
18
          sx pending rd req mask or
                                                                                                           var400_next_sx_pending_rd_sx_sel = sx_pending_rd_sx_sel;
19
          sx_pending_rd_sp_id or
                                                                                                           19 var400_next_sx_pending_rd_sp_id = sx_pending_rd_sp_id;
20
          sx pending rd state var indx or
                                                                                                           20 var400 next sx pending rd aux sel = sx pending rd aux sel;
21
         sx pending rd sx sel or
                                                                                                           21 var400 next sx pending rd aux inc = sx pending rd aux inc;
22
                                                                                                           22 var400_next_sx_pending_rd_pci = sx_pending_rd_pci;
         sx receive indx or
23
                                                                                                           23
                                                                                                                var400_next_sx_pending_rd_req_mask = sx_pending_rd_req_mask;
         sx to pa empty or
24
                                                                                                           24
                                                                                                                var400\_next\_sx\_pending\_rd\_state\_var\_indx = sx\_pending\_rd\_state\_var\_indx;
         sx_to_pa_vector[0] or
25
          sx_to_pa_vector[1] or
                                                                                                                var400 ignore this cycle = 'h0;
                                       Page 35 of 48
                                                                                                                                                  Page 36 of 48
                                                            Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                        Ex. 2117 - pa_ccg_sxifsm.v
```

```
1
       var400 next sx receive indx = sx receive indx;
                                                                                                                                 next_sx_pending_rd_sp_id = sx_pending_rd_sp_id;
 2
                                                                                                                            2 next_sx_pending_rd_aux_sel = sx_pending_rd_aux_sel;
 3
      // defaults
                                                                                                                                 next_sx_pending_rd_aux_inc = sx_pending_rd_aux_inc;
 4 next_sx_receive_indx = sx_receive_indx;
                                                                                                                                 next_sx_pending_rd_pci = sx_pending_rd_pci;
       next_sx_pending_advance = sx_pending_advance;
                                                                                                                                 next_sx_pending_rd_req_mask = sx_pending_rd_req_mask;
       sx_pending_fifo_advanceread = 'h0;
                                                                                                                                  next\_sx\_pending\_rd\_state\_var\_indx = sx\_pending\_rd\_state\_var\_indx;
       decrement\_pasx\_req\_cnt[0] = \text{'h0};
       decrement_pasx_req_cnt[1] = 'h0;
                                                                                                                                  var400_ignore_this_cycle = 'h0;
                                                                                                                                 if (sx_receive_indx == VECTORS_PER_SX_REQUEST) begin
       tcl_scratch_mem_point_write = 'h0;
       tcl_scratch_mem_position_write = 'h0;
                                                                                                                                  if (sx\_pending\_fifo\_empty == 'h0') begin \\
11
       next_position_address = position_address;
                                                                                                                                    var400_next_sx_pending_rd_sx_sel = initial_sx_pending_rd_sx_sel;
12     next_point_address = point_address;
                                                                                                                                    var400_next_sx_pending_rd_sp_id = initial_sx_pending_rd_sp_id;
13 decrement_available_positions = 'h0;
                                                                                                                                    var400 next sx pending rd aux sel = initial sx pending rd aux sel;
14 vertex_fifo_write = 'h0;
                                                                                                                                    var400 next sx pending rd aux inc = initial sx pending rd aux inc;
15 vertex fifo wr param cache indx = 'h0;
                                                                                                                           15
                                                                                                                                    var400 next sx pending rd pci = initial sx pending rd pci;
       vertex fifo wr state var indx = 'h0;
16
                                                                                                                           16
                                                                                                                                    var400 next sx pending rd req mask = initial sx pending rd req mask;
17
                                                                                                                           17
     vertex fifo wr edge flag = 'h0:
                                                                                                                                    var400 next sx pending rd state var indx = initial sx pending rd state var indx;
18
      vertex fifo wr kill flag = 'h0:
                                                                                                                           18
19
       sx to pa advanceread[0] = 'h0;
                                                                                                                           19
                                                                                                                                    var400 next sx receive indx = 'h0;
20
       sx to pa advanceread[1] = 'h0;
                                                                                                                           20
                                                                                                                                    next sx pending advance = 'h1;
21 //tcl_scratch_mem_position_data = sx_to_pa_vector[0];
                                                                                                                           21
                                                                                                                                    end
22 //tcl_scratch_mem_point_data
23 tcl_scratch_mem_position_data[VECTORX_HIGH:VECTORX_LOW];
                                                                                                                           22
                                                                                                                                   else begin
                                                                                                                           23
                                                                                                                                    var400_ignore_this_cycle = "h1;
24 tcl_scratch_mem_position_data = 'h0;
                                                                                                                           24
       tcl scratch mem point data = 'h0;
                                                                                                                           25
       next\_sx\_pending\_rd\_sx\_sel = sx\_pending\_rd\_sx\_sel;
                                             Page 37 of 48
                                                                                                                                                                        Page 38 of 48
                                                                     Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                                                 Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                     next sx pending advance = 'h0;
                                                                                                                            2
 2
       if (var400 ignore this cycle == 'h0) begin
                                                                                                                                    end
        var400_sx_sel = var400_next_sx_pending_rd_sx_sel;
                                                                                                                            3
                                                                                                                                   end
 4
                                                                                                                            4
         var400 vector write
                                                                                                                                   if ((sx_to_pa_empty[var400_sx_sel] == 'h1) ||
      var400_next_sx_pending_rd_req_mask[var400_next_sx_receive_indx[1:0]];
                                                                                                                                    ((var400_first_vector_write == 'h1) && (available_positions == 'h0)) ||
                                                                                                                                     ((var400_last_vector_write == 'h1) && (vertex_fifo_full == 'h1))) begin
        var400_first_vector_write = var400_vector_write & ~var400_next_sx_pending_rd_aux_sel;
                                                                                                                                    // do nothing this clock
        var400_last_vector_write = var400_vector_write &
      (var400_next_sx_pending_rd_aux_sel var400_next_sx_pending_rd_aux_inc);
                                                                                                                           10
                                                                                                                                   else begin
12
                                                                                                                           11
                                                                                                                                    sx_to_pa_advanceread[var400_sx_sel] = 'h1;
13
        if (next_sx_pending_advance == 'h1) begin
                                                                                                                           12
                                                                                                                           13
                                                                                                                                    if (var400_vector_write == 'h1) begin
14
         var400 vector wr cnt = 'h0;
                                                                                                                           14
15
         for (indx400 = 0; indx400 < VECTORS_PER_SX_REQUEST; indx400=indx400+1) begin
                                                                                                                           15
                                                                                                                                     vertex_fifo_wr_edge_flag = 'h0;
16
          if (var400 next sx pending rd req mask[indx400] == 'h1) begin
17
                                                                                                                                     vertex_fifo_wr_kill_flag = 'h0;
           var400 vector wr cnt = var400 vector wr cnt + 'h1;
                                                                                                                           17
18
          end
                                                                                                                           18
                                                                                                                                     if (var400 next sx pending rd aux sel == 'h1) begin
19
         end
                                                                                                                                      if (statevar bits use vtx point size == 'h1) begin
20
21
         if\left((sx\_to\_pa\_empty[var400\_sx\_sel] == 'h0\right) \&\&
                                                                                                                           20
                                                                                                                                        tcl scratch mem point write = 'h1;
                                                                                                                           21
                                                                                                                                        tcl scratch_mem_position_data = sx_to_pa_vector[var400_sx_sel];
22
23
           ((var400\_first\_vector\_write == 'h0) \ \| \ (var400\_vector\_wr\_cnt <= \ available\_positions))
                                                                                                                               tcl_scratch_mem_point_data tcl_scratch_mem_position_data[VECTORX_HIGH:VECTORX_LOW];
      ((var400_last_vector_write vertex_fifo_entriesavailable))) begin
                                                 'h0) || (var400_vector_wr_cnt
                                                                                                                           23
24
25
                                                                                                                           24
26
          sx_pending_fifo_advanceread = 'h1;
                                                                                                                           25
           decrement_pasx_req_cnt[var400_sx_sel] = 'h1;
                                                                                                                           26
                                                                                                                                       var400_sx_to_pa_vector_var400_sx_sel = sx_to_pa_vector[var400_sx_sel];
                                             Page 39 of 48
                                                                                                                                                                        Page 40 of 48
                                                                     Ex. 2117 - pa ccg sxifsm.v
                                                                                                                                                                                                 Ex. 2117 - pa ccg sxifsm.v
```

```
var400\_sx\_to\_pa\_vector\_y\_var400\_sx\_sel\\var400\_sx\_to\_pa\_vector\_var400\_sx\_sel[VECTORY\_HIGH:VECTORY\_LOW];
                                                                                                                                next_point_address = next_point_address + 'h1;
     var400\_sx\_to\_pa\_vector\_z\_var400\_sx\_sel\\var400\_sx\_to\_pa\_vector\_var400\_sx\_sel[VECTORZ\_HIGH:VECTORZ\_LOW];
                                                                                                                                vertex\_fifo\_wr\_param\_cache\_indx = \{var400\_next\_sx\_pending\_rd\_sp\_id,
                                                                                                                                                   var400 sx sel,
           if (statevar_bits_use_vtx_edge_flag == 'h1) begin
                                                                                                                                                   var400_next_sx_receive_indx[1:0],
            vertex_fifo_wr_edge_flag = var400_sx_to_pa_vector_y_var400_sx_sel[0];
                                                                                                                                                   var400_next_sx_pending_rd_pci};
                                                                                                                                vertex_fifo_wr_state_var_indx = var400_next_sx_pending_rd_state_var_indx;
                                                                                                                                vertex_fifo_write = 'h1;
10
           vertex fifo wr kill flag = 'h0;
                                                                                                                     10
11
           if (statevar bits use vtx kill flag == 'h1) begin
                                                                                                                     11
                                                                                                                               var400_next_sx_pending_rd_req_mask[var400_next_sx_receive_indx[1:0]] = 'h0:
12
            if (var400_sx_to_pa_vector_z_var400_sx_sel[30:0] != 'h0) begin
                                                                                                                     12
13
             vertex fifo wr kill flag = 'h1;
                                                                                                                     13
14
            end
                                                                                                                     14
                                                                                                                              var400_next_sx_receive_indx = var400_next_sx_receive_indx + 'h1;
15
           end
                                                                                                                     15
                                                                                                                            end
16
                                                                                                                     16
17
           end
                                                                                                                     17
18
          else begin
                                                                                                                     18
                                                                                                                           next sx pending rd sx sel = var400 next sx pending rd sx sel;
19
           tcl\_scratch\_mem\_position\_write = 'h1;
20
           next_position_address = next_position_address + 'h1;
                                                                                                                     19
                                                                                                                           next_sx_pending_rd_sp_id = var400_next_sx_pending_rd_sp_id;
21
           tcl_scratch_mem_position_data = sx_to_pa_vector[var400_sx_sel];
                                                                                                                     20
                                                                                                                           next_sx_pending_rd_aux_sel = var400_next_sx_pending_rd_aux_sel;
22
           decrement_available_positions = 'h1;
                                                                                                                     21
                                                                                                                          next_sx_pending_rd_aux_inc = var400_next_sx_pending_rd_aux_inc;
23
                                                                                                                     22
                                                                                                                           next_sx_pending_rd_pci = var400_next_sx_pending_rd_pci;
24
                                                                                                                           next_sx_pending_rd_req_mask = var400_next_sx_pending_rd_req_mask;
25
          if ((var400_next_sx_pending_rd_aux_sel == 'h1) \parallel
                                                                                                                           next\_sx\_pending\_rd\_state\_var\_indx = var400\_next\_sx\_pending\_rd\_state\_var\_indx;
                                                                                                                           next_sx_receive_indx = var400_next_sx_receive_indx;
            (var400_next_sx_pending_rd_aux_inc == 'h1)) begin
                                           Page 41 of 48
                                                                                                                                                                Page 42 of 48
                                                                  Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                                       Ex. 2117 - pa_ccg_sxifsm.v
 2
                                                                                                                         4 // pa to sx output register
 5 // sx pending rd
                                                                                                                      6 always @(posedge clk) begin: proc477
     always @(posedge clk) begin : proc475
                                                                                                                           if (reset == 'h1) begin
                                                                                                                            for (indx477 = 0: indx477 < PASX_INTERFACES: indx477 = indx477 + 1) begin
      if (reset == 'h1) begin
       sx_pending_rd_sx_sel = 'h0;
                                                                                                                             pa_to_sx_write[indx477] <= 'h0;
10
       sx_pending_rd_sp_id = 'h0;
                                                                                                                     10
                                                                                                                             pa\_to\_sx\_req[indx477] \mathrel{<=} 'h0;
                                                                                                                             pa_to_sx_sp_id[indx477] <= 'h0;
11
       sx_pending_rd_aux_sel = 'h0;
                                                                                                                     11
12
       sx_pending_rd_aux_inc = 'h0;
                                                                                                                     12
                                                                                                                             pa_to_sx_offset[indx477] <= 'h0;
13
       sx_pending_rd_pci = 'h0;
                                                                                                                             pa_to_sx_aux[indx477] <= 'h0;
14
       sx_pending_rd_req_mask = 'h0;
                                                                                                                     14
                                                                                                                             pa_to_sx_last[indx477] <= 'h0;
15
                                                                                                                     15
       sx_pending_rd_state_var_indx = 'h0;
                                                                                                                     16
16
17 else begin
                                                                                                                     17
18
      sx pending rd sx sel = next sx pending rd sx sel;
                                                                                                                     18
                                                                                                                             for (indx477 = 0; indx477 < PASX_INTERFACES; indx477 = indx477 + 1) begin
19
       sx_pending_rd_sp_id = next_sx_pending_rd_sp_id;
                                                                                                                             pa to sx write[indx477] <= next pa to sx write[indx477];
20
       sx pending rd aux sel = next sx pending rd aux sel;
                                                                                                                     20
                                                                                                                             pa to sx reg[indx477] <= next pa to sx reg[indx477];
21
                                                                                                                     21
       sx_pending_rd_aux_inc = next_sx_pending_rd_aux_inc;
                                                                                                                             pa to sx sp id[indx477] <= next pa to sx sp id[indx477];
22
                                                                                                                     22
                                                                                                                             pa to sx offset[indx477] <= next pa to sx offset[indx477];
       sx_pending_rd_pci = next_sx_pending_rd_pci;
23
                                                                                                                     23
       sx\_pending\_rd\_req\_mask = next\_sx\_pending\_rd\_req\_mask;
                                                                                                                             pa to sx aux[indx477] <= next pa to sx aux[indx477];
24
                                                                                                                     24
       sx_pending_rd_state_var_indx = next_sx_pending_rd_state_var_indx;
                                                                                                                             pa_to_sx_last[indx477] <= next_pa_to_sx_last[indx477];
25
                                                                                                                     25
                                          Page 43 of 48
                                                                                                                                                                Page 44 of 48
                                                                  Ex. 2117 - pa ccg sxifsm.v
                                                                                                                                                                                       Ex. 2117 - pa ccg sxifsm.v
```

```
end
                                                                                                                  1
                                                                                                                        end
 2
     end
                                                                                                                 2
                                                                                                                       if (decrement\_pasx\_req\_cnt[1] == \verb"h1") begin
 var480_next_pasx_req_cnt_1 = var480_next_pasx_req_cnt_1 - 'h1;
 5 // pasx_req_cnt
 always @(posedge clk) begin : proc480
                                                                                                                       pasx_req_cnt[0] <= var480_next_pasx_req_cnt_0;
      if (reset == 'h1) begin
                                                                                                                       pasx_req_cnt[1] <= var480_next_pasx_req_cnt_1;
       pasx_req_cnt[0] <= 'h0;
                                                                                                                 10
       pasx\_req\_cnt[1] \mathrel{<=} 'h0;
11
                                                                                                                     12
13
                                                                                                                 13 // registers
       var480 next pasx req cnt 0 = pasx req cnt[0];
       var480_next_pasx_req_cnt_l = pasx_req_cnt[1];
14
15
                                                                                                                 15 always @(posedge clk) begin : proc500
       if (increment_pasx_req_cnt[0] == 'h1) begin
                                                                                                                     if (reset == 'h1) begin
16
                                                                                                                16
17
                                                                                                                17
        var480\_next\_pasx\_req\_cnt\_0 = var480\_next\_pasx\_req\_cnt\_0 + 'h1;
                                                                                                                       sx request indx <= 'h0:
18
                                                                                                                18
                                                                                                                       aux sel <= 'h0:
19
                                                                                                                19
                                                                                                                       param cache base <= 'h0;
       if \, (decrement\_pasx\_req\_cnt[0] == \hbox{\tt '}h1) \ begin
                                                                                                                       sx aux <= 'h0:
20
                                                                                                                20
21
        var480_next_pasx_req_cnt_0 = var480_next_pasx_req_cnt_0 - 'h1;
                                                                                                                21
                                                                                                                       sx sent[0] <= 'h0;
22
                                                                                                                22
                                                                                                                       sx sent[1] <= 'h0;
23
                                                                                                                23
24
       if (increment\_pasx\_req\_cnt[1] == 'h1) \ begin
                                                                                                                24
                                                                                                                        sx_receive_indx <= VECTORS_PER_SX_REQUEST;
25
        var480\_next\_pasx\_req\_cnt\_l = var480\_next\_pasx\_req\_cnt\_l + 'h1;
                                                                                                                25
                                                                                                                        sx_pending_advance <= 'h0;
                                         Page 45 of 48
                                                                                                                                                         Page 46 of 48
                                                               Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                                Ex. 2117 - pa_ccg_sxifsm.v
       position_address <= 'h0;
2
       point address <= 'h0;
                                                                                                                 2
                                                                                                                      else begin
                                                                                                                        var600_vertex_fifo_entriesavailable = vertex_fifo_entriesavailable;
4
      else begin
       sx\_request\_indx <= next\_sx\_request\_indx;
                                                                                                                        if (vertex_fifo_write == 'h1) begin
                                                                                                                         var600_vertex_fifo_entriesavailable = var600_vertex_fifo_entriesavailable - 'h1;
       aux_sel <= next_aux_sel;
                                                                                                                 7
       param\_cache\_base <= next\_param\_cache\_base;
       sx aux <= next sx aux;
                                                                                                                 8
       sx\_sent[0] \le next\_sx\_sent[0];
                                                                                                                        if (vertex_fifo_advanceread == 'h1) begin
10
       sx\_sent[1] \mathrel{<=} next\_sx\_sent[1];
                                                                                                                10
                                                                                                                         var600\_vertex\_fifo\_entries available = var600\_vertex\_fifo\_entries available + "h1";
11
                                                                                                                11
12
       sx\_receive\_indx <= next\_sx\_receive\_indx;
                                                                                                                12
13
       sx_pending_advance <= next_sx_pending_advance;
                                                                                                                        vertex_fifo_entriesavailable <= var600_vertex_fifo_entriesavailable;
14
       position_address <= next_position_address;
                                                                                                                14
       point_address <= next_point_address;
                                                                                                                15
                                                                                                                16
17
                                                                                                                     endmodule
18
19
     // copy part of the control to the vertex fifo to get
20
21
    // the number of entries available
22
     23
    always @(posedge clk) begin : proc600
24
      if (reset == 'h1) begin
       vertex_fifo_entriesavailable <= VERTEX_FIFO_DEPTH + 1;
                                         Page 47 of 48
                                                                                                                                                         Page 48 of 48
                                                               Ex. 2117 - pa_ccg_sxifsm.v
                                                                                                                                                                                Ex. 2117 - pa_ccg_sxifsm.v
```

```
1 'include "header.v"
                                                                                                    1 //
                                                                                                                year of creation of the work.
2 'include "sc header.v"
 4 // Filename : sc.v
                                                                                                    4 module sc (
 5 // Description : SC top block
                                                                                                    6 // Chip Signals
 6 // Author : Mike Mantor
 7 // Created On : Sunday March 17 2002
 8 // Last Modified By: .
                                                                                                    8
                                                                                                         sclk global,
10 // Update Count : 0
                                                                                                    10
                                                                                                          RBBM_SC_soft_reset,
11 // Status : Initial
                                                                                                    11
                                                                                                         ROM SP0 disable.
                                                                                                    12 ROM_SP1_disable,
13 //---
                                                                                                    13 ROM_SP2_disable,
14 //
                                                                                                    14
                                                                                                          ROM SP3 disable,
15 // $Id: //depot/r400/devel/parts_lib/src/gfx/sc/sc.v#50 $
                                                                                                    15
16 //
                                                                                                    16
17 // $Change: 43702 $
                                                                                                    17
                                                                                                         RBBM_regclk_active, // Clock gating intiator
18 //
                                                                                                    18
19 //
20 // Copyright: Trade secret of ATI Technologies, Inc.
                                                                                                    20 // Interface to the Global Register Bus (RBBM)
21 //
              © Copyright 2001-2002, ATI Technologies, Inc., (unpublished)
                                                                                                    21 // ----
                                                                                                    22
23 //
              All rights reserved. This notice is intended as a precaution against
                                                                                                    23 RBBM we,
                                                                                                                            // write enable
24 //
                                                                                                   24 RBBM_wd,
                                                                                                                            // write data
              inadvertent publication and does not imply publication or any waiver
              of confidentiality. The year included in the foregoing notice is the
                                                                                                    25 RBBM_re,
                                    Page 1 of 87
                                                                                                                                        Page 2 of 87
                                                                                                                                                                      Ex. 2118 - sc.v
                                                                 Ex. 2118 - sc.v
 1 RBB_rs_in,
                     // read strobe daisy chain in
                                                                                                    1 // -----
                     // read strobe daisy chain out
     RBB rs out,
                                                                                                    2 PA SC p0,
      RBB_rd_in,
                       // read data daisy chain in
                                                                                                          PA_SC_p1,
      RBB_rd_out,
                       // read data daisy chain out
                                                                                                    5 PA SC p3,
                                                                                                    6 PA_SC_p4,
7 SC_RBBM_cntx0_busy, //
                                                                                                    7 PA_SC_xy0,
     SC_RBBM_cntx17_busy, //
                                                                                                    8 PA_SC_xy1,
                                                                                                          PA SC xy2,
10
      SC_a,
                                                                                                    10
11
     SC we,
                     // write enable
                                                                                                    11 PA SC cntl,
12 SC_wd,
                     // write data
                                                                                                    12 PA_SC_phase,
13 SC_re,
                  // read enable
                                                                                                    13 PA_SC_v0_indx,
14
                                                                                                    14 PA SC valid,
15
                                                                                                    15
      // Interface to the CP
                                                                                                         SC PA earlyfrz,
                                                                                                    16
17 CP SC wc inc. //Increment write confirm counter
                                                                                                   17 // Interface to the render central coarse tile
18 SC CP wc dec, //Decrement write confirm counter
                                                                                                   18 // ---
                                                                                                    19 SC_RC_coarse_event, //need to determine what field will carry id when its an event
19 SC_CP_vq_snd, //VisQuerry send visibility flag
20 SC_CP_vq_index, //VisQuerry index to identify one of 64 VQ flags
                                                                                                   20 SC_RC_coarse_tilex,
21 SC_CP_vq_discard, //VisQuerry 0=>keep, 1=>Discard
                                                                                                   21 SC RC coarse tiley.
      SC_CP_mp_snd,
                          //Sc is sending the multi-pass command now
                                                                                                          SC_RC_coarse_minz,
23
     SC_CP_mp_loop,
                          //0=>Continue 1=>Loop
                                                                                                    23 SC_RC_coarse_maxz,
                                                                                                    24 SC RC coarse zplane,
24
25 // Interface to the PA Setup Unit
                                                                                                    25 SC_RC_coarse_mask,
                                    Page 3 of 87
                                                                                                                                        Page 4 of 87
                                                                  Ex. 2118 - sc.v
                                                                                                                                                                      Ex. 2118 - sc.v
```

ATI 2118 LG v. ATI IPR2015-00326

```
SC RC coarse back,
                                                                                                    u0 SC SX quad tilex,
 2
                                                                                                2 u0_SC_SX_quad_tiley,
     SC RC coarse state,
                                                                                                     u0_SC_SX_quad_rb_id,
      SC RC coarse send,
      RC_SC_coarse_rtr,
                                                                                                     u0_SC_SX_quad_split,
     SC_RC_coarse_covered,
                                                                                                     u0_SC_SX_quad_send,
                                                                                                     u0_SX_SC_quad_rtr,
      // Interface to the render central hier mask
                                                                                                     // Interface to the shader export block SX1
      RC_SC_hier_mask,
      RC_SC_hier_rb_id,
                                                                                                10 u1_SC_SX_quad_x,
11
      RC_SC_hier_split,
                                                                                               11 u1_SC_SX_quad_y,
                                                                                               12 u1_SC_SX_quad_mask,
12
      RC_SC_hier_send,
13
      SC_RC_hier_rtr,
                                                                                               13 u1_SC_SX_quad_tilex,
                                                                                               14 u1_SC_SX_quad_tiley,
15
      // Interface to the render central detailed mask
                                                                                               15 u1_SC_SX_quad_rb_id,
                                                                                                    u1_SC_SX_quad_split,
16
17
      SC RC detail mask,
                                                                                               17 u1 SC SX quad send,
      SC RC detail send,
18
                                                                                               18 u1_SX_SC_quad_rtr,
19
      RC SC detail rtr,
                                                                                               19
20
                                                                                               20
                                                                                                    // Interface to the shader sequencer
21
     // Interface to the shader export block SX0
                                                                                               21 // -----
                                                                                               22 SC_SQ_data, //
22
23
     u0 SC SX quad x,
                                                                                               23
                                                                                                    SC SQ valid,
24
     u0_SC_SX_quad_y,
                                                                                               24
                                                                                                    SQ_SC_free_buff, //
25
      u0_SC_SX_quad_mask,
                                                                                                    SQ_SC_dec_cntr_cnt, //
                                    Page 5 of 87
                                                                                                                                   Page 6 of 87
                                                                Ex. 2118 - sc.v
                                                                                                                                                               Ex. 2118 - sc.v
                                                                                                     2
 3
      // Interface to shader pipes
                                                                                                      // I/O Definitions
                                                                                                      4
     u0_SC_SP_data,
                                                                                                     // Chip Signals
      u0_SC_SP_type,
                                                                                                     input
                                                                                                             sclk global;
      u0_SC_SP_last_quad,
                                                                                                7
                                                                                                     input
      u0_SC_SP_valid,
                                                                                                             RBBM SC soft reset;
                                                                                                     input
                                                                                                      input ROM_SP0_disable;
10
      ul SC SP data,
                                                                                                10
                                                                                                     input
                                                                                                             ROM_SP1_disable;
11
      ul_SC_SP_type,
                                                                                               11
                                                                                                              ROM_SP2_disable;
12
      u1_SC_SP_last_quad,
                                                                                               12
                                                                                                              ROM_SP3_disable;
13
      ul_SC_SP_valid,
                                                                                               13
14
                                                                                               14
                                                                                                             CG_SC_pm_enb;
15
     u2_SC_SP_data,
                                                                                               15
                                                                                                             RBBM_regclk_active; // Clock gating intiator
     u2_SC_SP_type,
17
     u2_SC_SP_last_quad,
                                                                                               17
                                                                                                    // Interface to the Register Bus (RBBM)
18
     u2_SC_SP_valid,
                                                                                               18
                                                                                                     input [16:2] RBBM_a;
19
                                                                                                     input RBBM_we;
     u3_SC_SP_data,
                                                                                                     input [31:0] RBBM_wd;
20
                                                                                               20
21
     u3_SC_SP_type,
                                                                                                    input RBBM_re;
                                                                                               21
22
     u3_SC_SP_last_quad,
                                                                                               22
                                                                                                    input RBB_rs_in;
23
     u3_SC_SP_valid
                                                                                               23
                                                                                                    output RBB_rs_out;
24
                                                                                               24
                                                                                                     input [31:0] RBB rd in;
                                                                                                     output [31:0] RBB_rd_out;
25
                                    Page 7 of 87
                                                                                                                                   Page 8 of 87
                                                               Ex. 2118 - sc.v
                                                                                                                                                               Ex. 2118 - sc.v
```

```
input [31:0] PA SC p0;
2
       output SC RBBM cntx0 busy;
                                                                                                                    input [39:0] PA SC p1;
3
       output
                SC RBBM cntx17 busy:
                                                                                                                     input [31:0] PA_SC_p2;
4
                                                                                                                     input [31:0] PA_SC_p3;
      // Registered out for possible daisy chaining of Rbbm bus
                                                                                                                     input [31:0] PA_SC_p4;
       output [16:2] SC_a;
                                                                                                                     input [13:0] PA_SC_zminmax;
       output SC_we;
                                                                                                                     input [29:0] PA_SC_cntl;
       output [31:0] SC_wd;
                                                                                                                     input [1:0] PA_SC_phase;
       output SC_re;
                                                                                                                     input [1:0] PA_SC_v0_indx;
                                                                                                                     input PA_SC_valid;
11
       // Interface to the CP
                                                                                                                    output SC_PA_earlyfrz;
12
13
      input CP_SC_wc_inc;
                                                                                                                    // Interface to Render Central Coarse Tile
                                     //Increment write confirm counter
14
      output SC CP we dec;
                                     //Decrement write confirm counter
15
      output SC CP vq snd;
                                     //VisQuerry send visibility flag
                                                                                                             15
                                                                                                                    output SC RC coarse event;
                                     //VisQuerry index to identify one of 64 VQ flags
                                                                                                                    output [9:0] SC RC coarse tilex;
16
       output [5:0] SC CP vq index;
                                                                                                             16
17
                                                                                                             17
       output SC CP vq discard; //VisQuerry 0=>keep, 1=>Discard
                                                                                                                    output [9:0] SC RC coarse tiley:
18
                SC CP_mp_snd;
                                      //Sc is sending the multi-pass command now
                                                                                                             18
                                                                                                                    output [13:0] SC RC coarse minz;
       output
                                    //0=>Continue 1=>Loop
19
       output
                SC_CP_mp_loop;
                                                                                                             19
                                                                                                                     output [13:0] SC RC coarse maxz;
20
                                                                                                             20
                                                                                                                    output [95:0] SC RC coarse zplane;
21
      // Interface to the PA Setup Unit
                                                                                                             21
                                                                                                                     output [15:0] SC RC coarse mask;
22
                                                                                                             22
                                                                                                                    output SC RC coarse back;
23
       input [17:0] PA SC xy0;
                                                                                                             23
                                                                                                                     output [2:0] SC_RC_coarse_state;
24
       input [17:0] PA_SC_xy1;
                                                                                                             24
                                                                                                                     output SC_RC_coarse_send;
25
       input [17:0] PA_SC_xy2;
                                                                                                                     output SC_RC_coarse_covered;
                                         Page 9 of 87
                                                                                                                                                      Page 10 of 87
                                                                         Ex. 2118 - sc.v
                                                                                                                                                                                      Ex. 2118 - sc.v
       input RC_SC_coarse_rtr;
                                                                                                                    output u0_SC_SX_quad_send;
2
                                                                                                                     input u0 SX SC quad rtr;
       // Interface to Render Central Hier Kill
4
                                                                                                                    // Interface to the shader export block SX1
       input [15:0] RC SC hier mask;
                                                                                                                     output [1:0] u1 SC SX quad x;
       input [1:0] RC_SC_hier_rb_id;
       input RC SC hier split;
                                                                                                                     output [1:0] u1_SC_SX_quad_y;
       input RC SC hier send;
                                                                                                                     output [31:0] u1_SC_SX_quad_mask;
       output SC_RC_hier_rtr;
                                                                                                                     output [1:0] u1_SC_SX_quad_tilex;
10
                                                                                                             10
                                                                                                                     output u1_SC_SX_quad_tiley;
11
       // Interface to Render Central Detail Quad Mask
                                                                                                             11
                                                                                                                     output [1:0] u1_SC_SX_quad_rb_id;
12
                                                                                                             12
                                                                                                                    output u1_SC_SX_quad_split;
13
       output [15:0] SC_RC_detail_mask;
                                                                                                                    output u1_SC_SX_quad_send;
14
       output SC_RC_detail_send;
                                                                                                                     input u1_SX_SC_quad_rtr;
15
       input RC_SC_detail_rtr;
16
                                                                                                                    // Interface to the shader sequencer
17
       // Interface to the shader export block SX0
                                                                                                             17
18
                                                                                                             18
                                                                                                                   input SQ_SC_free_buff; //used to synchronize SP and SQ interface
19
       output [1:0] u0_SC_SX_quad_x;
                                                                                                                   input SQ_SC_dec_cntr_cnt; //to prevent SQ request queue from overflowing
                                                                                                                     output [`SC_SQ_DATA_WIDTH-1:0] SC_SQ_data; //This data bus has a 1st clk and
20
       output [1:0] u0 SC SX quad y;

    20 output [`SC_S
    21 2nd clock definiti

21
      output [31:0] u0 SC SX quad mask;
                                                                                                                              SC_SQ_valid; //Data at the output is valid and assumed sampled at next
                                                                                                                     output
22
      output [1:0] u0 SC SX quad tilex;
                                                                                                             23
                                                                                                                  clock
23
      output u0_SC_SX_quad_tiley;
                                                                                                             24
24
      output [1:0] u0 SC SX quad rb id;
                                                                                                             25
                                                                                                                    // Interface to the shader pipe SP0
       output u0 SC SX quad split;
                                        Page 11 of 87
                                                                                                                                                      Page 12 of 87
                                                                         Ex. 2118 - sc.v
                                                                                                                                                                                      Ex. 2118 - sc.v
```

```
output [99:0] u0_SC_SP_data;
 2
       output [1:0] u0_SC_SP_type;
                                                                                                                 output u0 SC SP last quad;
       output u0_SC_SP_valid;
                                                                                                                 // Internal Signal definitions
                                                                                                                 // Interface to the shader pipe SP1
       output [99:0] u1_SC_SP_data;
       output [1:0] u1_SC_SP_type;
                                                                                                                          sclk_reg;
       output ul_SC_SP_last_quad;
                                                                                                                          sclk_sc;
11
      output ul_SC_SP_valid;
                                                                                                          11
12
                                                                                                          12
                                                                                                                wire
                                                                                                                          sc_srst;
13
      // Interface to the shader pipe SP2
                                                                                                          13
                                                                                                                wire
                                                                                                                          se hard srst;
14
                                                                                                          14
                                                                                                                          sc_soft_srst;
                                                                                                                 wire
15
       output [99:0] u2_SC_SP_data;
                                                                                                          15
16
       output [1:0] u2 SC SP type;
                                                                                                          16
                                                                                                                          cg blk gated clk override;
                                                                                                                 wire
17
       output u2_SC_SP_last_quad;
                                                                                                          17
18
       output u2 SC SP valid;
                                                                                                          18
                                                                                                                          regclk active:
                                                                                                                 wire
19
                                                                                                          19
                                                                                                                 wire
                                                                                                                          reg_clk_en;
20
      // Interface to the shader pipe SP3
                                                                                                          20
21
                                                                                                          21
                                                                                                                 wire
                                                                                                                         se clk en;
22
       output [99:0] u3 SC SP data;
                                                                                                          22
23
       output [1:0] u3_SC_SP_type;
                                                                                                          23
                                                                                                                 wire [16:2] reg_a;
24
       output u3_SC_SP_last_quad;
                                                                                                          24
       output u3_SC_SP_valid;
                                                                                                                 wire [31:0] reg_wd;
                                       Page 13 of 87
                                                                                                                                                 Page 14 of 87
                                                                       Ex. 2118 - sc.v
                                                                                                                                                                                Ex. 2118 - sc.v
                                                                                                                        st jss enable;
                                                                                                                 wire
       wire
                reg re;
 2
                                                                                                                 wire [1:0] st jss x dim;
                                                                                                                 wire [1:0] st_jss_y_dim;
 3
       wire
                SC TS;
 4
       wire [31:0] sc rd:
                                                                                                                 wire [3:0] st_max_sample_dist;
 5
                                                                                                                 wire [3:0] st_jss_sample0_sel;
       //wire
                su earlyfrz;
                                                                                                                 wire [3:0] st_jss_sample1_sel;
 7
                                                                                                                 wire [3:0] st_jss_sample2_sel;
                                                                                                                 wire [3:0] st_jss_sample3_sel;
                st_msaa_enable;
                                                                                                                 wire [3:0] st_jss_sample4_sel;
10
       wire [3:0] st_msaa_num_samples;
                                                                                                          10
                                                                                                                 wire [3:0] st_jss_sample5_sel;
11
                                                                                                          11
                                                                                                                 wire [3:0] st_jss_sample6_sel;
12
                st_draw_zero_length_line;
                                                                                                          12
                                                                                                                wire [3:0] st_jss_sample7_sel;
13
               st_window_offset_disable;
                                                                                                                 wire [3:0] st_jss_sample8_sel;
14
       wire [13:0] st_window_scissor_x_min;
                                                                                                                wire [3:0] st_jss_sample9_sel;
15
       wire [13:0] st_window_scissor_x_max;
                                                                                                                wire [3:0] st_jss_sample10_sel;
       wire [13:0] st_window_scissor_y_min;
                                                                                                                wire [3:0] st_jss_sample11_sel;
17
       wire [13:0] st_window_scissor_y_max;
                                                                                                          17
                                                                                                                wire [3:0] st_jss_sample12_sel;
18
       wire [14:0] st_x_offset;
                                                                                                          18
                                                                                                                wire [3:0] st_jss_sample13_sel;
19
       wire [14:0] st_y_offset;
                                                                                                                wire [3:0] st_jss_sample14_sel;
20
                                                                                                          20
                                                                                                                wire [3:0] st jss sample15 sel;
      wire st bres cntl en;
21
      wire [7:0] st bres cntl reg;
                                                                                                          21
                                                                                                                wire [3:0] st msaa urc samp offset x;
22
                                                                                                          22
                                                                                                               wire [3:0] st msaa urc samp offset y;
23
                                                                                                          23
      wire [31:0] st aa mask;
                                                                                                                wire [3:0] st msaa llc samp offset x;
24
                                                                                                          24
      wire st_output_screen_xy;
                                                                                                                wire [3:0] st_msaa_llc_samp_offset_y;
25
       wire
               st_line_stipple_enable;
                                                                                                                wire [3:0] st_msaa_lrc_samp_offset_x;
                                       Page 15 of 87
                                                                                                                                                 Page 16 of 87
                                                                      Ex. 2118 - sc.v
                                                                                                                                                                                Ex. 2118 - sc.v
```

```
wire [3:0] st_msaa_lrc_samp_offset_y;
                                                                                                                        wire [3:0] st_sample_12_x;
2
       wire [3:0] st sample 0 x;
                                                                                                                        wire [3:0] st_sample_12_y;
       wire [3:0] st_sample_0_y;
                                                                                                                        wire [3:0] st_sample_13_x;
       wire [3:0] st_sample_1_x;
                                                                                                                        wire [3:0] st_sample_13_y;
       wire [3:0] st_sample_1_y;
                                                                                                                        wire [3:0] st_sample_14_x;
       wire [3:0] st_sample_2_x;
                                                                                                                        wire [3:0] st_sample_14_y;
       wire [3:0] st_sample_2_y;
                                                                                                                        wire [3:0] st_sample_15_x;
       wire [3:0] st_sample_3_x;
                                                                                                                        wire [3:0] st_sample_15_y;
                                                                                                                  9 // wire [127:0] st_aa_offset_tbl;
       wire [3:0] st_sample_3_y;
       wire [3:0] st_sample_4_x;
                                                                                                                       wire [15:0] st_line_pattern;
11
       wire [3:0] st_sample_4_y;
                                                                                                                11
                                                                                                                       wire [7:0] st_repeat_count;
12
       wire [3:0] st_sample_5_x;
                                                                                                                12
                                                                                                                       wire [3:0] st_pattern_start;
13
       wire [3:0] st_sample_5_y;
                                                                                                                13
                                                                                                                       wire st pattern bit order;
14
       wire [3:0] st_sample_6_x;
                                                                                                                                st auto reset enable;
15
       wire [3:0] st sample 6 y;
                                                                                                                15
                                                                                                                       wire [3:0] st current ptr;
16
       wire [3:0] st sample 7 x;
                                                                                                                16
                                                                                                                       wire [7:0] st current count;
17
       wire [3:0] st_sample_7_y;
                                                                                                                17
                                                                                                                       wire st cliprect enable;
18
       wire [3:0] st_sample_8_x;
                                                                                                                18
                                                                                                                       wire [13:0] st_cliprect_0_x_min;
19
       wire [3:0] st_sample_8_y;
                                                                                                                19
                                                                                                                       wire [13:0] st_cliprect_0_y_min;
20
       wire [3:0] st_sample_9_x;
                                                                                                                20
                                                                                                                       wire [13:0] st_cliprect_0_x_max;
21
       wire [3:0] st sample 9 y;
                                                                                                                21
                                                                                                                       wire [13:0] st cliprect 0 y max;
22
       wire [3:0] st sample 10 x;
                                                                                                                22
                                                                                                                       wire [13:0] st cliprect 1 x min;
23
       wire [3:0] st_sample_10_y;
                                                                                                                23
                                                                                                                       wire [13:0] st_cliprect_1_y_min;
24
       wire [3:0] st_sample_11_x;
                                                                                                                24
                                                                                                                        wire [13:0] st_cliprect_1_x_max;
25
       wire [3:0] st_sample_11_y;
                                                                                                                25
                                                                                                                        wire [13:0] st_cliprect_1_y_max;
                                         Page 17 of 87
                                                                                                                                                          Page 18 of 87
                                                                           Ex. 2118 - sc.v
                                                                                                                                                                                           Ex. 2118 - sc.v
       wire [13:0] st_cliprect_2_x_min;
                                                                                                                        wire [13:0] zminmax_in;
2
                                                                                                                        wire [31:0] p0 in;
       wire [13:0] st cliprect 2 y min;
       wire [13:0] st_cliprect_2_x_max;
                                                                                                                        wire [39:0] p1 in;
4
       wire [13:0] st_cliprect_2_y_max;
                                                                                                                        wire [31:0] p2 in;
                                                                                                                        wire [31:0] p3_in;
       wire [13:0] st_cliprect_3_x_min;
                                                                                                                        wire [31:0] p4 in;
       wire [13:0] st_cliprect_3_y_min;
       wire [13:0] st_cliprect_3_x_max;
                                                                                                                                 valid in:
       wire [13:0] st_cliprect_3_y_max;
                                                                                                                        wire [1:0] v0 indx in;
       wire [15:0] st_clip_rule;
                                                                                                                        wire [1:0] phase_in;
10
       wire st_poly_offset_front_enable;
                                                                                                                10
                                                                                                                                 event_in;
11
               st_poly_offset_back_enable;
                                                                                                                11
                                                                                                                        wire [3:0] event_id_in;
12
                st_poly_offset_para_enable;
                                                                                                                12
                                                                                                                        wire [17:0] xy0_in;
13
       wire [31:0] st_poly_offset_front_offset;
                                                                                                                        wire [17:0] xy1_in;
14
       wire [31:0] st_poly_offset_back_offset;
                                                                                                                14
                                                                                                                        wire [17:0] xy2_in;
15
                                                                                                                15
       wire [31:0] st_poly_offset_front_scale;
16
                                                                                                                        wire [31:0] RBIU_wdata; // Register Write Data
       wire [31:0] st_poly_offset_back_scale;
17
                                                                                                                17
                                                                                                                                  RBIU_we;
18
                                                                                                                18
                                                                                                                                  RBIU_re;
                st iter msaa enable;
19
       wire [3:0] st_iter_msaa_num_samples;
                                                                                                                19
                                                                                                                       wire [2:0] RBIU_waddr;
20
                                                                                                                       wire [2:0] RBIU raddr;
       wire st iter iss enable;
                                                                                                                20
21
                                                                                                                21
                                                                                                                                 RBIU cpv;
                                                                                                                       wire
22
                                                                                                                22
                                                                                                                                 RBIU PIPE_susc_cntl_sel;
       wire [270:0] pa sc inputs;
                                                                                                                       wire
23
                                                                                                                23
       wire [270:0] pa sc inputs reg;
                                                                                                                                 RBIU PIPE_sq_context_misc_sel;
                                                                                                                       wire
24
                                                                                                                24
                                                                                                                                  RBIU PIPE window offset sel;
                                                                                                                       wire
25
       wire [29:0] cntl in;
                                                                                                                25
                                                                                                                       wire
                                                                                                                                 RBIU_PIPE_aa_config_sel;
                                         Page 19 of 87
                                                                                                                                                          Page 20 of 87
                                                                          Ex. 2118 - sc.v
                                                                                                                                                                                           Ex. 2118 - sc.v
```

```
1
       wire
                 RBIU PIPE aa mask sel;
                                                                                                                1
                                                                                                                      wire
                                                                                                                                RBIU_PIPE_poly_offset_front_offset_sel;
2
       wire
                 RBIU_PIPE_jss_sample_sel_0_sel;
                                                                                                                2
                                                                                                                      wire
                                                                                                                                RBIU\_PIPE\_poly\_offset\_back\_scale\_sel;
 3
       wire
                 RBIU_PIPE_jss_sample_sel_l_sel;
                                                                                                                3
                                                                                                                      wire
                                                                                                                                RBIU\_PIPE\_poly\_offset\_back\_offset\_sel;
4 // wire
                  RBIU_PIPE_msaa_2x2_offset_sel;
                                                                                                                4
                                                                                                                      wire [31:0] PIPE RBIU rdata;
 5 // wire
                  RBIU_PIPE_aa_offset_tbl_0_sel;
 6 // wire
                  RBIU_PIPE_aa_offset_tbl_l_sel;
                                                                                                                      wire
                                                                                                                                sr_prim_we;
 7 // wire
                  RBIU_PIPE_aa_offset_tbl_2_sel;
                                                                                                                               sr_z_we;
 8 // wire
                  RBIU_PIPE_aa_offset_tbl_3_sel;
                                                                                                                               sr_pipe_valid;
                 RBIU_PIPE_line_stipple_sel;
                                                                                                                                sr_event;
10
                 RBIU_PIPE_line_stipple_state_sel;
                                                                                                               10
                                                                                                                      wire [3:0] sr_event_id;
11
                 RBIU_PIPE_line_cntl_sel;
                                                                                                               11
                                                                                                                               sr_null_prim;
                                                                                                                      wire
12
                 RBIU_PIPE_window_scissor_tl_sel;
                                                                                                               12
                                                                                                                     wire [2:0] sr_dealloc_slot;
13
                 RBIU_PIPE_window_scissor_br_sel;
                                                                                                               13
       wire
                                                                                                                     wire
                                                                                                                             sr first prim of slot;
14
                 RBIU_PIPE_screen_scissor_tl_sel;
                                                                                                               14
                                                                                                                              sr_end_of_pkt;
       wire
                                                                                                                     wire
15
                 RBIU PIPE screen scissor br sel;
                                                                                                               15
                                                                                                                              sr back face;
                                                                                                                     wire
       wire
                 RBIU_PIPE_cliprect_0_tl_sel;
16
                                                                                                                     wire [1:0] sr provoking vertex;
       wire
                                                                                                               16
17
                 RBIU PIPE cliprect 0 br sel:
                                                                                                               17
       wire
                                                                                                                     wire sr x major:
18
                 RBIU PIPE cliprect 1 tl sel:
                                                                                                               18
                                                                                                                              sr start in diamond:
       wire
                                                                                                                     wire
19
       wire
                 RBIU PIPE cliprect 1 br sel;
                                                                                                               19
                                                                                                                     wire
                                                                                                                              sr end in diamond;
                 RBIU_PIPE_cliprect_2_tl_sel;
                                                                                                                     wire [2:0] sr_prim_type;
20
       wire
                                                                                                               20
                 RBIU PIPE cliprect 2 br sel;
21
       wire
                                                                                                              21
                                                                                                                     wire [1:0] sr phase;
                 RBIU PIPE cliprect 3 tl sel;
22
       wire
                                                                                                              22
                                                                                                                     wire [2:0] sr state var indx;
23
       wire
                 RBIU PIPE cliprect 3 br sel;
                                                                                                              23
                                                                                                                     wire [2:0] qpp_state_var_indx;
24
       wire
                 RBIU_PIPE_cliprect_rule_sel;
                                                                                                              24
                                                                                                                     wire [17:0] sr_v0;
25
                 RBIU_PIPE_poly_offset_front_scale_sel;
                                                                                                               25
                                                                                                                      wire [17:0] sr_v1;
                                        Page 21 of 87
                                                                                                                                                       Page 22 of 87
                                                                          Ex. 2118 - sc.v
                                                                                                                                                                                         Ex. 2118 - sc.v
       wire [17:0] sr_v2;
                                                                                                                      wire [39:0] sr_zy_zff;
2
       wire [17:0] sr ref x;
                                                                                                                2
                                                                                                                      wire [13:0] sr z min zff;
       wire [17:0] sr_ref_y;
                                                                                                                      wire [13:0] sr_z_max_zff;
       wire [31:0] sr i0:
                                                                                                                4
                                                                                                                      wire
                                                                                                                               sr cntx0 busy;
       wire [31:0] sr ix;
                                                                                                                      wire
                                                                                                                               sr_cntx1to7_busy;
       wire [31:0] sr iy;
       wire [31:0] sr_j0;
                                                                                                                7
                                                                                                                      wire
                                                                                                                                prim ff re;
       wire [31:0] sr_jx;
                                                                                                                8
                                                                                                                      wire
                                                                                                                                prim ff full;
       wire [31:0] sr_jy;
                                                                                                                9
                                                                                                                                prim_ff_empty;
10
       wire [31:0] sr_w0;
                                                                                                               10
                                                                                                                      wire ['SC_PRIM_INTERP_WIDTH -1:0] prim_ff_wr_data;
11
       wire [31:0] sr_wx;
                                                                                                               11
                                                                                                                      wire [\SC_PRIM_INTERP_WIDTH -1:0] prim_ff_rd_data;
12
       wire [31:0] sr_wy;
                                                                                                               12
                                                                                                                               z_ff_re;
13
                                                                                                               13
                                                                                                                               z_ff_full;
       wire [10:0] sr_param_cache_indx0
14
                                                                                                               14
       wire [10:0] sr_param_cache_indx1;
                                                                                                                               z_ff_empty;
15
                                                                                                                      wire ['SC_ZDATA_WIDTH -1:0] z_ff_wr_data;
       wire [10:0] sr_param_cache_indx2;
                                                                                                                     wire ['SC_ZDATA_WIDTH -1:0] z_ff_rd_data;
              sr_null_prim_zff;
17
                                                                                                               17
                                                                                                                             tile_ff_re;
               sr_zy_max_zff;
18
               sr back face zff;
                                                                                                               18
                                                                                                                               tile_ff_full;
19
       wire [2:0] sr_prim_type_zff;
                                                                                                                              tile_ff_empty;
                                                                                                                     wire ['SC_TILEDATA_WIDTH -1:0] tile_ff_wr_data;
20
      wire sr polymode zff;
                                                                                                               20
                                                                                                              21 // wire [171:0] tile_ff_rd_data;
21
      wire [2:0] sr state var indx zff;
22
                                                                                                               22
      wire [17:0] sr ref x zff:
23
       wire [17:0] sr ref v zff:
                                                                                                               23
                                                                                                                     wire
                                                                                                                                pipe rts;
24
       wire [31:0] sr z0 zff;
                                                                                                               24
                                                                                                                     wire [1:0] pipe_phase;
       wire [39:0] sr zx zff;
                                                                                                               25
                                                                                                                     wire
                                                                                                                               event flag;
                                         Page 23 of 87
                                                                                                                                                       Page 24 of 87
                                                                          Ex. 2118 - sc.v
                                                                                                                                                                                         Ex. 2118 - sc.v
```

```
wire [3:0] event id;
                                                                                                               wire [36:0] cw e0;
2
       wire [36:0] e0;
                                                                                                              wire [31:0] cw_e0y;
       wire [31:0] e0 y;
                                                                                                               wire [18:0] cw dxe0;
       wire [18:0] e0_dx;
                                                                                                               wire [18:0] cw_dye0;
       wire [18:0] e0_dy;
                                                                                                               wire [36:0] cw_e1;
       wire [36:0] e1;
                                                                                                               wire [31:0] cw_e1x;
       wire [31:0] e1_x;
                                                                                                               wire [18:0] cw_dxe1;
       wire [18:0] e1_dx;
                                                                                                               wire [18:0] cw_dye1;
       wire [18:0] e1_dy;
                                                                                                               wire [36:0] cw_e2;
                                                                                                               wire [31:0] cw_e2x;
11
      wire [31:0] e2_x;
                                                                                                              wire [18:0] cw_dxe2;
12
      wire [18:0] e2_dx;
                                                                                                        12
                                                                                                              wire [18:0] cw_dye2;
13
      wire [18:0] e2_dy;
                                                                                                        13
                                                                                                              wire cw xdir;
14
                                                                                                              wire [9:0] cw_tilex;
      wire x dir;
15
      wire [12:0] x start;
                                                                                                        15
                                                                                                              wire [2:0] cw xmin;
16
      wire [12:0] x end;
                                                                                                        16
                                                                                                              wire [2:0] cw xmax;
17
                                                                                                        17
      wire y_dir;
                                                                                                              wire cw ydir;
18
      wire [12:0] y_start;
                                                                                                        18
                                                                                                              wire
                                                                                                                       cw xmajor;
19
       wire [12:0] y end;
                                                                                                        19
                                                                                                              wire [9:0] cw tiley;
20
      wire [3:0] bb fract bits;
                                                                                                        20
                                                                                                              wire [2:0] cw ymin;
21
     wire pass_empty_prim;
                                                                                                        21
                                                                                                              wire [2:0] cw ymax;
22
      wire
              cw event;
                                                                                                        22
                                                                                                              wire cw last tile;
23
      wire [3:0] cw_event_id;
                                                                                                        23
                                                                                                              wire
                                                                                                                        cw_tile_valid;
24
      wire [3:0] cw_bb_fract_bits;
                                                                                                        24
                                                                                                              wire
                                                                                                                        pipe_freeze_b_early;
25
      wire cw_pass_empty_prim;
                                                                                                        25
                                                                                                                        pipe_freeze_b_dly;
                                      Page 25 of 87
                                                                                                                                              Page 26 of 87
                                                                     Ex. 2118 - sc.v
                                                                                                                                                                              Ex. 2118 - sc.v
                                                                                                               wire [18:0] qm_dye2;
                pipe freeze b dly1;
       wire
2
                                                                                                         2
      wire
                qmsk_z_freeze_b;
                                                                                                               wire [15:0] qm_quadmask;
3
                                                                                                         3
       wire
                qm last tile;
                                                                                                                       qm_quadmask_valid;
4
                                                                                                         4
       wire
                qm_event;
       wire [3:0] qm_event_id;
                                                                                                         5
                                                                                                               wire [`SC_PRIM_INTERP_WIDTH -1:0] qpp_prim_data;
      wire
                qm_xdir;
                                                                                                         6
                                                                                                               wire qpp_fpos_early;
7
                                                                                                         7
                qm_ydir;
                                                                                                                      qpp_last_qdpair_of_prim;
       wire [9:0] qm_tilex;
                                                                                                                       qpp_q0_last_of_tile;
       wire [9:0] qm_tiley;
                                                                                                                       qpp_q0_zmask_needed;
                                                                                                                        qpp_q0_qhit;
10
       wire [1:0] qm_tilex_m3;
                                                                                                        10
11
       wire [1:0] qm_tiley_m3;
                                                                                                        11
                                                                                                               wire [9:0] qpp_q0_tilex;
12
       wire [2:0] qm_xmin;
                                                                                                        12
                                                                                                               wire [9:0] qpp_q0_tiley;
13
       wire [2:0] qm_xmax;
                                                                                                               wire [1:0] qpp_q0_quadx;
                                                                                                              wire [1:0] qpp_q0_quady;
14
       wire [2:0] qm_ymin;
                                                                                                        14
15
                                                                                                        15
      wire [2:0] qm_ymax;
                                                                                                              wire [1:0] qpp_q0_rb_id;
      wire [3:0] qm_bb_fract_bits;
                                                                                                                      qpp_q0_split;
17
     wire qm_z_mask_needed;
                                                                                                        17
                                                                                                              wire [7:0] qpp_q0_ulc_sample_mask;
18
      wire [36:0] qm_e0;
                                                                                                        18
                                                                                                              wire [7:0] qpp q0 urc sample mask;
19
      wire [36:0] qm_e1;
                                                                                                              wire [7:0] qpp_q0_llc_sample_mask;
20
      wire [36:0] qm e2;
                                                                                                        20
                                                                                                              wire [7:0] qpp q0 lrc sample mask;
21
     wire [18:0] qm dxe0;
                                                                                                        21
                                                                                                              wire [2:0] qpp_q0_ulc_cntrmost_sample_id;
22
     wire [18:0] am dve0:
                                                                                                        22
                                                                                                              wire [2:0] qpp_q0_urc_cntrmost_sample_id;
23
      wire [18:0] am dxe1:
                                                                                                        23
                                                                                                              wire [2:0] qpp_q0_llc_cntrmost_sample_id;
24
      wire [18:0] qm dye1;
                                                                                                        24
                                                                                                              wire [2:0] qpp_q0_lrc_cntrmost_sample_id;
       wire [18:0] qm_dxe2;
                                                                                                        2.5
                                                                                                              wire qpp_ql_last_of_tile;
                                      Page 27 of 87
                                                                                                                                              Page 28 of 87
                                                                     Ex. 2118 - sc.v
                                                                                                                                                                             Ex. 2118 - sc.v
```

```
wire [13:0] zff z min;
 1
       wire
                 qpp_q1_zmask_needed;
2
       wire
                 qpp_q1_qhit;
                                                                                                                       wire [13:0] zff z max;
3
       wire [9:0] qpp_q1_tilex;
                                                                                                                       wire [31:0] zff z0;
       wire [9:0] qpp_q1_tiley;
                                                                                                                       wire [39:0] zff_zx;
       wire [1:0] qpp_ql_quadx;
                                                                                                                       wire [39:0] zff_zy;
       wire [1:0] qpp_q1_rb_id;
               qpp_q1_split;
                                                                                                                             rc_event;
       wire [1:0] qpp_q1_quady;
                                                                                                                       wire [9:0] rc_tilex;
       wire [7:0] qpp_ql_ulc_sample_mask;
                                                                                                                       wire [9:0] rc_tiley;
                                                                                                                       wire [13:0] rc_minz;
       wire [7:0] qpp_q1_urc_sample_mask;
11
       wire [7:0] qpp_q1_llc_sample_mask;
                                                                                                               11
                                                                                                                       wire [13:0] rc_maxz;
12
       wire [7:0] qpp_ql_lrc_sample_mask;
                                                                                                               12
                                                                                                                       wire [95:0] rc_zplane;
13
                                                                                                               13
                                                                                                                      wire [15:0] rc_mask;
       wire [2:0] qpp q1 ulc cntrmost sample id;
14
                                                                                                                                rc_back;
       wire [2:0] qpp q1 urc entrmost sample id:
                                                                                                                      wire
15
       wire [2:0] gpp q1 llc entrmost sample id;
                                                                                                               15
                                                                                                                      wire [2:0] rc state;
16
       wire [2:0] gpp q1 lrc cntrmost sample id;
                                                                                                               16
                                                                                                                      wire
                                                                                                                               rc send;
17
                                                                                                               17
                                                                                                                     wire
                                                                                                                                rc covered:
18
                zff null prim:
                                                                                                               18
       wire
                                                                                                                     wire
                                                                                                                               re rtr;
19
       wire
                zff zy max;
                                                                                                               19
                                                                                                                      wire
                                                                                                                                re hier rtr;
                                                                                                                      wire [15:0] rc_in_hier_mask;
20
       wire
                zff back face;
                                                                                                               20
21
      wire [2:0] zff prim type;
                                                                                                               21
                                                                                                                      wire [1:0] rc in rb id;
22
      wire zff polymode;
                                                                                                               22
                                                                                                                     wire
                                                                                                                                rc in split;
23
       wire [2:0] zff_state_var_indx;
                                                                                                               23
                                                                                                                      wire
                                                                                                                                rc_in_hier_send;
24
       wire [17:0] zff_ref_x;
                                                                                                               24
       wire [17:0] zff_ref_y;
                                                                                                               25
                                                                                                                                qdpkr_in_fz;
                                         Page 29 of 87
                                                                                                                                                        Page 30 of 87
                                                                          Ex. 2118 - sc.v
                                                                                                                                                                                         Ex. 2118 - sc.v
                                                                                                                       wire [2:0] pkr_sv_indx;
                                                                                                                2
2
       //Signals for the sc rc detailed z interface
3
                                                                                                                3
       wire
                 detail mask accum rdy;
                                                                                                                       wire
                                                                                                                               pkr cntx0 busy;
4
                 detail hit 0:
                                                                                                                4
       wire
                                                                                                                       wire
                                                                                                                                pkr cntx1to7 busy;
       wire
                 detail lqt 0:
                                                                                                                       wire
                                                                                                                                pkr_iter_cntx0_busy;
       wire
                 detail hit 1;
                                                                                                                       wire
                                                                                                                                pkr_iter_cntx1to7_busy;
                 detail_lqt_1;
                                                                                                                7
       wire [15:0] detail mask;
                                                                                                                8
                                                                                                                      wire
                                                                                                                                iterator input fz;
                detail_mask_valid;
                                                                                                                       wire [2:0] iterator_sv_indx;
                                                                                                                       wire [1:0] iterator_SX0_quad_x;
10
                 rdy_for_detail_mask;
                                                                                                               10
11
                                                                                                               11
                                                                                                                       wire [1:0] iterator_SX0_quad_y;
12
       //Signals between the packer and iterator
                                                                                                               12
                                                                                                                       wire [31:0] iterator_SX0_quad_mask;
13
       wire [`SC_QD_DATA_WIDTH-1:0] pkr_qd0;
                                                                                                               13
                                                                                                                       wire [1:0] iterator_SX0_quad_tilex;
14
       wire [`SC_QD_DATA_WIDTH-1:0] pkr_qd1;
                                                                                                               14
                                                                                                                                iterator_SX0_quad_tiley;
15
       wire [`SC_QD_DATA_WIDTH-1:0] pkr_qd2;
                                                                                                               15
                                                                                                                     wire [1:0] iterator_SX0_quad_rb_id;
       wire [`SC_QD_DATA_WIDTH-1:0] pkr_qd3;
                                                                                                                               iterator_SX0_quad_split;
17
                pkr_qdhit0;
                                                                                                               17
                                                                                                                             iterator_SX0_quad_send;
18
                 pkr qdhit1;
                                                                                                               18
                                                                                                                               SX0_iterator_quad_rtr;
19
                                                                                                                      wire [1:0] iterator_SX1_quad_x;
                pkr qdhit2;
20
       wire
               pkr qdhit3;
                                                                                                               20
                                                                                                                     wire [1:0] iterator SX1 quad v;
      wire [`SC_PRIM_INTERP_WIDTH -1:0] pkr_primdata;
21
                                                                                                               21
                                                                                                                      wire [31:0] iterator SX1 quad mask;
22
                                                                                                               22
                                                                                                                     wire [1:0] iterator SX1 quad tilex:
      wire
               pkr ds one clk command;
23
                                                                                                               23
                                                                                                                                iterator SX1 quad tilev:
      wire
               pkr ds end of prim;
                                                                                                                     wire
24
                                                                                                               24
      wire
               pkr_ds_end_of_vector;
                                                                                                                     wire [1:0] iterator SX1 quad rb id;
25
       wire
               pkr send row;
                                                                                                               25
                                                                                                                     wire iterator SX1 quad split;
                                        Page 31 of 87
                                                                                                                                                        Page 32 of 87
                                                                         Ex. 2118 - sc.v
                                                                                                                                                                                         Ex. 2118 - sc.v
```

```
iterator SX1 quad send;
       wire
2
       wire
                 SX1_iterator_quad_rtr;
       wire
                 SQ iterator free buff;
                 SQ_iterator_dec_cntr_cnt;
       wire ['SC_SQ_DATA_WIDTH-1:0] iterator_SQ_data;
                 iterator_SQ_valid;
       wire [99:0] iterator_SP0_data;
                                                                                                                          ati_master_clock_permanent uati_master_clock_permanent(
       wire [1:0] iterator_SP0_type;
                                                                                                                           .clk_in(sclk_global),
                 iterator_SP0_last_quad;
10
                 iterator_SP0_valid;
11
       wire [99:0] iterator_SP1_data;
12
       wire [1:0] iterator_SP1_type;
                                                                                                                          ati_dff_in #(1) uati_dff_in_pm_en(
13
                iterator_SP1_last_quad;
                                                                                                                           .clk(sclk),
14
                 iterator_SP1_valid;
                                                                                                                           .d(CG_SC_pm_enb),
                                                                                                                  15
15
       wire [99:0] iterator SP2 data;
                                                                                                                           .q(cg_blk_gated_clk_override)
16
       wire [1:0] iterator SP2 type;
                                                                                                                  16
17
               iterator SP2 last quad:
                                                                                                                  17
       wire
18
                 iterator SP2 valid:
                                                                                                                  18
                                                                                                                         //create clock enable signals based on active signals
       wire
19
       wire [99:0] iterator SP3 data;
                                                                                                                  19
                                                                                                                         ati dff in #(1) uati dff in regclk active(
       wire [1:0] iterator SP3 type;
20
                                                                                                                  20
                                                                                                                           clk(sclk)
                                                                                                                  21
                                                                                                                           .d(RBBM_regclk_active),
21
                iterator SP3 last quad;
22
                                                                                                                  22
                                                                                                                           .q(regclk active)
23
       wire
                 rt_set_cntx0_busy;
                                                                                                                  23
24
                 cntx0_decr;
                                                                                                                  24
                 cntx1to7_decr;
                                                                                                                         //This enable would be intiated by rbbm_regclk_active and held high in the
                                          Page 33 of 87
                                                                                                                                                            Page 34 of 87
                                                                            Ex. 2118 - sc.v
                                                                                                                                                                                              Ex. 2118 - sc.v
       //block as long as necessary to ensure all data could be read
                                                                                                                   2
2
       assign reg clk en = regclk active | SC RBBM cntx0 busy | SC RBBM cntx17 busy;
                                                                                                                         //RBBM Interface register
4
       //This active signal would be a collection of request from external blocks that require
                                                                                                                         ati rbbm intf uati rbbm intf(
5
       //the block clocks to be enabled along with internal busy signals that require the clocks
                                                                                                                            .sclk_reg(sclk_reg),
                                                                                                                            .rbbm we(RBBM we),
     .rbbm_re(RBBM_re),
                                                                                                                            .rbbm_a(RBBM_a),
                                                                                                                            .rbbm\_wd(RBBM\_wd),
10
                                                                                                                   10
                                                                                                                            .reg_we(reg_we),
11
       //Generate the sclk_reg clock tree
                                                                                                                  11
                                                                                                                            .reg_re(reg_re),
12
       ati_master_clock_gater uati_master_clock_gater_sclk_reg (
                                                                                                                            .reg_a(reg_a),
13
         .clk_in(sclk_global),
                                                                                                                            .reg_wd(reg_wd),
14
                                                                                                                           .pipe_we(SC_we),
15
         .en(reg_clk_en),
                                                                                                                           .pipe_re(SC_re),
16
         .pm_enb(cg_blk_gated_clk_override),
                                                                                                                            .pipe_a(SC_a),
17
         .clk out(sclk reg)
                                                                                                                            .pipe_wd(SC_wd),
18
                                                                                                                            .rbbm_rs_in(RBB_rs_in),
19
                                                                                                                           .rbbm_rd_in(RBB_rd_in),
20
       //Generate sclk sc clock tree
                                                                                                                  20
                                                                                                                           .block rs(sc rs),
21
       ati_master_clock_gater uati_master_clock_gater_sc_clk (
                                                                                                                  21
                                                                                                                           .block rd(sc rd),
22
         .clk in(sclk_global),
                                                                                                                  22
                                                                                                                           .rbbm_rs_out(RBB_rs_out),
23
         clk(sclk)
                                                                                                                  23
                                                                                                                           .rbbm_rd_out(RBB_rd_out)
24
         .en(sc clk en),
                                                                                                                  24
25
          .pm_enb(cg_blk_gated_clk_override),
                                                                                                                  25
         .clk out(sclk sc)
                                         Page 35 of 87
                                                                                                                                                            Page 36 of 87
                                                                            Ex. 2118 - sc.v
                                                                                                                                                                                              Ex. 2118 - sc.v
```

1		1	.q(SC_PA_earlyfrz)	
2	//register input reset	2);	
3	ati_dff_in #(1) uati_dff_in_sc_hard_srst(3		
4	.clk(sclk),	4		
5	.d(srst),	5		
6	.q(sc_hard_srst)	6		
7);	7		
8		8	//	
9	//register input soft resets	9	// Register inputs	
10	ati_dff_in #(1) uati_dff_in_sc_soft_srst(10	//	
11	.clk(sclk),	11		
12	.d(RBBM_SC_soft_reset),	12	sc_interface_regs usc_interface_regs(
13	.q(sc_soft_srst)	13	.clk(sclk_sc),	
14);	14	.en(pipe_freeze_b_dly),	
15		15	.pa_sc_inputs(pa_sc_inputs),	
16	//use this in the block for the srst everywhere except	16	.pa_sc_inputs_reg(pa_sc_inputs_reg),	
17	//state storage that can only get reset by a hard reset	17	.RC_SC_coarse_rtr(RC_SC_coarse_rtr),	
18	assign sc_srst = sc_soft_srst sc_hard_srst;	18	.rc_rtr(rc_rtr),	
19		19	.RC_SC_hier_send(RC_SC_hier_send),	
20		20	.rc_in_hier_send(rc_in_hier_send),	
21	//register block outputs using the ati_dff_out or ati_dff_en_out	21	.RC_SC_hier_mask(RC_SC_hier_mask),	
22	ati_dff_out #(1) uati_dff_out_earlyfrz(22	.rc_in_hier_mask(rc_in_hier_mask),	
23	.clk(sclk_sc),	23	.RC_SC_hier_rb_id(RC_SC_hier_rb_id),	
24	//.d(su_earlyfrz),	24	.rc_in_rb_id(rc_in_rb_id),	
25	.d(pipe_freeze_b_early),	25	.RC_SC_hier_split(RC_SC_hier_split),	
	Page 37 of 87		Page 38 of 87	
	Ex. 21	18 - sc.v		Ex. 2118 - sc.v
1	so in solitos in solit)		no estata/no estata)	
1	.re_in_split(re_in_split),	1	.rc_state(rc_state),	
2	$.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr),$	2	$.SC_RC_coarse_state(SC_RC_coarse_state),$	
2	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr),	2 3	.SC_RC_coarse_state(SC_RC_coarse_state), .rc_covered(rc_covered),	
2 3 4	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr),	2 3 4	.SC_RC_coarse_state(SC_RC_coarse_state), .rc_covered(rc_covered), .SC_RC_coarse_covered(SC_RC_coarse_covered),	
2 3 4 5	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr),	2 3 4 5	.SC_RC_coarse_state(SC_RC_coarse_state), .rc_covered(rc_covered), .SC_RC_coarse_covered(SC_RC_coarse_covered), .rc_send(rc_send),	
2 3 4 5 6	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff),	2 3 4 5 6	.SC_RC_coarse_state(SC_RC_coarse_state), .rc_covered(rc_covered), .SC_RC_coarse_covered(SC_RC_coarse_covered), .rc_send(rc_send), .SC_RC_coarse_send(SC_RC_coarse_send),	
2 3 4 5 6 7	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(U1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff),	2 3 4 5 6	.SC_RC_coarse_state(SC_RC_coarse_state), .rc_covered(rc_covered), .SC_RC_coarse_covered(SC_RC_coarse_covered), .rc_send(rc_send), .SC_RC_coarse_send(SC_RC_coarse_send), .rc_hier_rtr(rc_hier_rtr),	
2 3 4 5 6 7 8	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_SC_dec_cntr_cnt(SQ_SC_dec_cntr_cnt),	2 3 4 5 6 7 8	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr),	
2 3 4 5 6 7 8	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(U1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_SC_dee_cntr_cnt(SQ_SC_dee_cntr_cnt), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt),	2 3 4 5 6 7 8	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid),	
2 3 4 5 6 7 8 9	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), .SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(SX0_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_SC_dee_cntr_cnt(SQ_SC_dee_cntr_cnt), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .re_event(rc_event),	2 3 4 5 6 7 8 9	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), re_hier_tr(rc_hier_tr), SC_RC_hier_tr(SC_RC_hier_tr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send),	
2 3 4 5 6 7 8 9 10	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(SX0_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_SC_dee_cntr_cnt), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event),	2 3 4 5 6 7 8 9	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_tr(rc_hier_tr), SC_RC_hier_tr(SC_RC_hier_tr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask),	
2 3 4 5 6 7 8 9 10 11	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(SX0_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_SC_dee_cntr_cnt), .SQ_SC_dee_cntr_cnt(SQ_SC_dee_cntr_cnt), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex),	2 3 4 5 6 7 8 9 10	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask),	
2 3 4 5 6 7 8 9 10 11 12 13	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_SC_dec_entr_ent), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex),	2 3 4 5 6 7 8 9 10 11 12 13	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_tr(rc_hier_tr), SC_RC_hier_tr(SC_RC_hier_tr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x),	
2 3 4 5 6 7 8 9 10 11 12 13	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_SC_dee_cntr_cnt), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .rc_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tiley(rc_tiley),	2 3 4 5 6 7 8 9 10 11 12 13	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SX0_quad_x(iterator_SX0_quad_x), u0_SC_SX_quad_x(u0_SC_SX_quad_x),	
2 3 4 5 6 7 8 9 10 11 12 13 14	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .rc_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tiley(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley),	2 3 4 5 6 7 8 9 10 11 12 13 14	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SX0_quad_x(iterator_SX0_quad_x), u0_SC_SX_quad_x(id_SC_SX_quad_x), iterator_SX0_quad_y(iterator_SX0_quad_x),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_SC_dec_cntr_cnt), .SQ_iterator_dec_cntr_cnt(SQ_iterator_dec_cntr_cnt), .rc_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tiley(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .rc_minz(rc_minz),	2 3 4 5 6 7 8 9 10 11 12 13 14 15	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), u0_SC_SX_quad_x(id_SC_SX_quad_x), iterator_SXO_quad_y(iterator_SXO_quad_y), u0_SC_SX_quad_y(iterator_SXO_quad_y),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .re_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .re_tilex(re_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .re_tiley(re_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .re_minz(re_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz),	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(rc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_x(id_SC_SC_quad_x), iterator_SXO_quad_y(id_SC_SX_quad_y), iterator_SXO_quad_y(id_SC_SX_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .rc_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tiley(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .rc_minz(rc_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .rc_maxz(rc_maxz),	2 3 3 4 5 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(sc_hier_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_x(iderator_SXO_quad_x), iterator_SXO_quad_y(iderator_SXO_quad_y), iterator_SXO_quad_y(iderator_SXO_quad_y), iterator_SXO_quad_y(iderator_SXO_quad_y), iterator_SXO_quad_mask(iderator_SXO_quad_mask), iterator_SXO_quad_mask(iderator_SXO_quad_mask), iterator_SXO_quad_mask(iderator_SXO_quad_mask),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .re_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .re_tilex(re_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .re_tiley(re_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .re_minz(re_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .re_maxz(rc_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz),	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(sender_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_v(iterator_SXO_quad_y), iterator_SXO_quad_y(iterator_SXO_quad_y), iterator_SXO_quad_w(iterator_SXO_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_tilex(iterator_SXO_quad_mask), iterator_SXO_quad_tilex(iterator_SXO_quad_mask),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .re_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .re_tilex(re_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .re_tiley(re_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .re_minz(re_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .re_maxz(re_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .re_zplane(re_zplane),	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(senier_tr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_x(uO_SC_SX_quad_x), iterator_SXO_quad_y(iterator_SXO_quad_y), iterator_SXO_quad_w(sc_SC_SX_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(uO_SC_SX_quad_mask), iterator_SXO_quad_mask(uO_SC_SX_quad_tilex), uO_SC_SX_quad_tilex(iterator_SXO_quad_tilex),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tilex(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .rc_minz(rc_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .rc_maxz(rc_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .rc_zplane(rc_zplane), .SC_RC_coarse_zplane(SC_RC_coarse_zplane),	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(senier_tr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_v(lo_SC_SX_quad_x), iterator_SXO_quad_y(lo_SC_SX_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(uO_SC_SX_quad_mask), iterator_SXO_quad_mask(uO_SC_SX_quad_mask), iterator_SXO_quad_tilex(iterator_SXO_quad_milex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .re_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .re_tilex(re_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .re_tiley(re_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .re_minz(re_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .re_maxz(re_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .re_zplane(re_zplane), .SC_RC_coarse_zplane(SC_RC_coarse_zplane), .re_mask(re_mask),	2 2 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(senier_tr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_x(uO_SC_SX_quad_x), iterator_SXO_quad_y(iterator_SXO_quad_y), iterator_SXO_quad_w(sc_SC_SX_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), uo_SC_SX_quad_tilex(iterator_SXO_quad_tilex),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dec_cntr_cnt(SQ_iterator_dec_cntr_cnt), .SQ_iterator_dec_cntr_cnt(SQ_iterator_dec_cntr_cnt), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tilex(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .rc_minz(rc_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .rc_maxz(rc_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .rc_zplane(rc_zplane), .SC_RC_coarse_zplane(SC_RC_coarse_zplane), .rc_mask(rc_mask), .SC_RC_coarse_mask(SC_RC_coarse_mask),	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(sender_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_v(lo_SC_SX_quad_x), iterator_SXO_quad_y(iterator_SXO_quad_y), iterator_SXO_quad_w(lo_SC_SX_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(u_SC_SX_quad_mask), iterator_SXO_quad_tilex(iterator_SXO_quad_milex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tiley(iterator_SXO_quad_tilex), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .SQ_iterator_dec_entr_ent(SQ_iterator_dec_entr_ent), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tilex(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .rc_minz(rc_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .rc_maxz(rc_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .rc_zplane(rc_zplane), .SC_RC_coarse_zplane(SC_RC_coarse_zplane), .rc_mask(rc_mask), .SC_RC_coarse_mask(SC_RC_coarse_mask), .rc_back(rc_back),	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(senier_tr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SX0_quad_x(iterator_SX0_quad_x), iterator_SX0_quad_x(u0_SC_SX_quad_x), iterator_SX0_quad_y(iterator_SX0_quad_y), id_SC_SX_quad_y(u0_SC_SX_quad_y), iterator_SX0_quad_mask(iterator_SX0_quad_mask), iterator_SX0_quad_mask(iterator_SX0_quad_mask), iterator_SX0_quad_tilex(iterator_SX0_quad_tilex), iterator_SX0_quad_tilex(iterator_SX0_quad_tilex), iterator_SX0_quad_tilex(iterator_SX0_quad_tilex), iterator_SX0_quad_tiley(iterator_SX0_quad_tiley), iterator_SX0_quad_tiley(iterator_SX0_quad_tiley), iterator_SX0_quad_rb_id(iterator_SX0_quad_rb_id),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), .SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dec_cntr_cnt(SQ_iterator_dec_cntr_cnt), .SQ_iterator_dec_cntr_cnt(SQ_iterator_dec_cntr_cnt), .rc_event(rc_event), .SC_RC_coarse_event(SC_RC_coarse_event), .rc_tilex(rc_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .rc_tilex(rc_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .rc_minz(rc_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .rc_maxz(rc_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .rc_zplane(rc_zplane), .SC_RC_coarse_zplane(SC_RC_coarse_zplane), .rc_mask(rc_mask), .SC_RC_coarse_mask(SC_RC_coarse_mask),	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(sender_rtr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SXO_quad_x(iterator_SXO_quad_x), iterator_SXO_quad_v(lo_SC_SX_quad_x), iterator_SXO_quad_y(iterator_SXO_quad_y), iterator_SXO_quad_w(lo_SC_SX_quad_y), iterator_SXO_quad_mask(iterator_SXO_quad_mask), iterator_SXO_quad_mask(u_SC_SX_quad_mask), iterator_SXO_quad_tilex(iterator_SXO_quad_milex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tilex(iterator_SXO_quad_tilex), iterator_SXO_quad_tiley(iterator_SXO_quad_tilex), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley), iterator_SXO_quad_tiley(iterator_SXO_quad_tiley),	
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.u0_SX_SC_quad_rtr(u0_SX_SC_quad_rtr), SX0_iterator_quad_rtr(SX0_iterator_quad_rtr), .u1_SX_SC_quad_rtr(u1_SX_SC_quad_rtr), SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), SX1_iterator_quad_rtr(SX1_iterator_quad_rtr), SQ_SC_free_buff(SQ_SC_free_buff), .SQ_iterator_free_buff(SQ_iterator_free_buff), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .SQ_iterator_dee_cntr_cnt(SQ_iterator_dee_cntr_cnt), .re_event(re_event), .SC_RC_coarse_event(SC_RC_coarse_event), .re_tilex(re_tilex), .SC_RC_coarse_tilex(SC_RC_coarse_tilex), .re_tilex(re_tiley), .SC_RC_coarse_tiley(SC_RC_coarse_tiley), .re_minz(re_minz), .SC_RC_coarse_minz(SC_RC_coarse_minz), .re_maxz(re_maxz), .SC_RC_coarse_maxz(SC_RC_coarse_maxz), .re_zplane(re_zplane), .SC_RC_coarse_zplane(SC_RC_coarse_zplane), .re_mask(re_mask), .SC_RC_coarse_mask(SC_RC_coarse_mask), .re_back(re_back), .SC_RC_coarse_back(SC_RC_coarse_back), .Page 39 of 87	2 3 3 4 5 6 6 7 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	SC_RC_coarse_state(SC_RC_coarse_state), rc_covered(rc_covered), SC_RC_coarse_covered(SC_RC_coarse_covered), rc_send(rc_send), SC_RC_coarse_send(SC_RC_coarse_send), rc_hier_rtr(senier_tr), SC_RC_hier_rtr(SC_RC_hier_rtr), detail_mask_valid(detail_mask_valid), SC_RC_detail_send(SC_RC_detail_send), detail_mask(detail_mask), SC_RC_detail_mask(SC_RC_detail_mask), iterator_SX0_quad_x(iterator_SX0_quad_x), iterator_SX0_quad_x(u0_SC_SX_quad_x), iterator_SX0_quad_y(iterator_SX0_quad_y), id_SC_SX_quad_y(u0_SC_SX_quad_y), iterator_SX0_quad_mask(iterator_SX0_quad_mask), iterator_SX0_quad_mask(iterator_SX0_quad_mask), iterator_SX0_quad_tilex(iterator_SX0_quad_tilex), iterator_SX0_quad_tilex(iterator_SX0_quad_tilex), iterator_SX0_quad_tilex(iterator_SX0_quad_tilex), iterator_SX0_quad_tiley(iterator_SX0_quad_tiley), iterator_SX0_quad_tiley(iterator_SX0_quad_tiley), iterator_SX0_quad_rb_id(iterator_SX0_quad_rb_id),	Ex. 2118 - sc.y

1	$. iterator_SX0_quad_split (iterator_SX0_quad_split),$		1	.iterator_SQ_valid(iterator_SQ_valid),
2	$.u0_SC_SX_quad_split(u0_SC_SX_quad_split),$		2	.SC_SQ_valid(SC_SQ_valid),
3			3	.iterator_SP0_data(iterator_SP0_data),
4	$. iterator_SX0_quad_send (iterator_SX0_quad_send),$		4	.u0_SC_SP_data(u0_SC_SP_data),
5	.u0_SC_SX_quad_send(u0_SC_SX_quad_send),		5	.iterator_SP0_type(iterator_SP0_type),
6	.iterator_SX1_quad_x(iterator_SX1_quad_x),		6	.u0_SC_SP_type(u0_SC_SP_type),
7	.ul_SC_SX_quad_x(ul_SC_SX_quad_x),		7	.iterator_SPO_last_quad(iterator_SPO_last_quad),
8	.iterator_SX1_quad_y(iterator_SX1_quad_y),		8	.u0_SC_SP_last_quad(u0_SC_SP_last_quad),
10	.ul_SC_SX_quad_y(ul_SC_SX_quad_y), .iterator_SX1_quad_mask(iterator_SX1_quad_mask),		10	.iterator_SP0_valid(iterator_SP0_valid), .u0_SC_SP_valid(u0_SC_SP_valid),
11	.ul_SC_SX_quad_mask(ul_SC_SX_quad_mask),		11	.iterator_SP1_data(iterator_SP1_data),
12	.iterator_SX1_quad_tilex(iterator_SX1_quad_tilex),		12	.ul_SC_SP_data(ul_SC_SP_data),
13	.ul_SC_SX_quad_tilex(ul_SC_SX_quad_tilex),		13	.iterator_SP1_type(iterator_SP1_type),
14	.iterator_SX1_quad_tiley(iterator_SX1_quad_tiley),		14	.ul_SC_SP_type(ul_SC_SP_type),
15	.u1_SC_SX_quad_tiley(u1_SC_SX_quad_tiley),		15	.iterator_SP1_last_quad(iterator_SP1_last_quad),
16			16	$.u1_SC_SP_last_quad(u1_SC_SP_last_quad),$
17	$. iterator_SX1_quad_rb_id (iterator_SX1_quad_rb_id),$		17	.iterator_SP1_valid(iterator_SP1_valid),
18	$.u1_SC_SX_quad_rb_id(u1_SC_SX_quad_rb_id),$		18	.u1_SC_SP_valid(u1_SC_SP_valid),
19	$. iterator_SX1_quad_split (iterator_SX1_quad_split),$		19	.iterator_SP2_data(iterator_SP2_data),
20	.u1_SC_SX_quad_split(u1_SC_SX_quad_split),		20	.u2_SC_SP_data(u2_SC_SP_data),
21			21	.iterator_SP2_type(iterator_SP2_type),
22	.iterator_SX1_quad_send(iterator_SX1_quad_send),		22	.u2_SC_SP_type(u2_SC_SP_type),
23	.u1_SC_SX_quad_send(u1_SC_SX_quad_send),		23	.iterator_SP2_last_quad(iterator_SP2_last_quad),
24	.iterator_SQ_data(iterator_SQ_data),		24	.u2_SC_SP_last_quad(u2_SC_SP_last_quad),
25	.SC_SQ_data(SC_SQ_data),		25	.iterator_SP2_valid(iterator_SP2_valid),
	Page 41 of 87	Ex. 2118 - sc.v		Page 42 of 87 Ex. 2118 - sc.v
		EX. 2116 - SC.V		EX. 2110 - SC.V
1	.u2_SC_SP_valid(u2_SC_SP_valid),		1	.iRBBM_we_ql(reg_we), // write enable
2	.iterator_SP3_data(iterator_SP3_data),		2	.iRBBM_wd_q1(reg_wd), // write data
2	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data),		2 3	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable
2 3 4	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type),		2 3 4	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain
2	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data),		2 3	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable
2 3 4 5	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type),		2 3 4 5	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out
2 3 4 5	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type), .iterator_SP3_last_quad(iterator_SP3_last_quad),		2 3 4 5 6	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(sc_rs), // read strobe daisy chain out .oRBIU_block_rd(sc_rd), // read data daisy chain out
2 3 4 5 6 7	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type), .iterator_SP3_last_quad(iterator_SP3_last_quad), .u3_SC_SP_last_quad(u3_SC_SP_last_quad),		2 3 4 5 6 7	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe
2 3 4 5 6 7 8	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type), .iterator_SP3_last_quad(iterator_SP3_last_quad), .u3_SC_SP_last_quad(u3_SC_SP_last_quad), .iterator_SP3_valid(iterator_SP3_valid),		2 3 4 5 6 7 8	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(sc_rs), // read strobe daisy chain out .oRBIU_block_rd(sc_rd), // read data daisy chain out // Interface to sc_pipe .oRBIU_we(RBIU_we),
2 3 4 5 6 7 8	iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type), .iterator_SP3_last_quad(iterator_SP3_last_quad), .u3_SC_SP_last_quad(u3_SC_SP_last_quad), .iterator_SP3_valid(iterator_SP3_valid), .u3_SC_SP_valid(u3_SC_SP_valid),		2 3 4 5 6 7 8	.iRBBM_wd_ql(reg_wd), // write data .iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(sc_rs), // read strobe daisy chain out .oRBIU_block_rd(sc_rd), // read data daisy chain out // Interface to sc_pipe .oRBIU_we(RBIU_we), .oRBIU_re(RBIU_re),
2 3 4 5 6 7 8 9 10 11	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type), .iterator_SP3_last_quad(iterator_SP3_last_quad), .u3_SC_SP_last_quad(iterator_SP3_last_quad), .iterator_SP3_valid(iterator_SP3_valid), .iterator_SP3_valid(iterator_SP3_valid), .u3_SC_SP_valid(u3_SC_SP_valid), .sr_cntx0_busy(sr_cntx0_busy), .SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), .sr_cntx1to7_busy(sr_cntx1to7_busy),		2 3 4 5 6 7 8 9 10 11	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(se_rs), // read strobe daisy chain out oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe oRBIU_we(RBIU_we), oRBIU_re(RBIU_re), oRBIU_wddr(RBIU_waddr), oRBIU_raddr(RBIU_raddr), oRBIU_wdata(RBIU_raddr), oRBIU_wdata(RBIU_wdata),
2 3 4 5 6 7 8 9 10 11 12	.iterator_SP3_data(iterator_SP3_data), .u3_SC_SP_data(u3_SC_SP_data), .iterator_SP3_type(iterator_SP3_type), .u3_SC_SP_type(u3_SC_SP_type), .iterator_SP3_last_quad(iterator_SP3_last_quad), .u3_SC_SP_last_quad(iterator_SP3_last_quad), .iterator_SP3_valid(iterator_SP3_valid), .u3_SC_SP_valid(iterator_SP3_valid), .u3_SC_SP_valid(u3_SC_SP_valid), .sr_entx0_busy(sr_entx0_busy), .SC_RBBM_entx0_busy(SC_RBBM_entx0_busy), .sr_entx1to7_busy(sr_entx1to7_busy), .SC_RBBM_entx17_busy(SC_RBBM_entx17_busy),		2 3 4 5 6 7 8 9 10 11 12	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(sc_rs), // read strobe daisy chain out oRBIU_block_rd(sc_rd), // read data daisy chain out // Interface to sc_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_reb), oRBIU_wddr(RBIU_raddr), oRBIU_wdata(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata),
2 3 4 5 6 7 8 9 10 11 12 13	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), sr_cntx1to7_busy(sr_cntx1to7_busy), SC_RBBM_cntx17_busy(SC_RBBM_cntx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr),		2 3 4 5 6 7 8 9 10 11 12 13	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(sc_rs), // read strobe daisy chain out oRBIU_block_rd(sc_rd), // read data daisy chain out // Interface to sc_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_reb), oRBIU_re(RBIU_reb), oRBIU_waddr(RBIU_raddr), oRBIU_wdata(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata), oRBIU_poy(RBIU_roy),
2 3 4 5 6 7 8 9 10 11 12 13 14	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(iterator_SP3_valid), sr_entx0_busy(sr_entx0_busy), SC_RBBM_entx0_busy(SC_RBBM_entx0_busy), sr_entx1to7_busy(sr_entx1to7_busy), SC_RBBM_entx17_busy(SC_RBBM_entx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)		2 3 4 5 6 7 8 9 10 11 12 13 14	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(sc_rs), // read strobe daisy chain out oRBIU_block_rd(sc_rd), // read data daisy chain out // Interface to sc_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_re), oRBIU_re(RBIU_re), oRBIU_waddr(RBIU_waddr), oRBIU_waddr(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata), oRBIU_copy(RBIU_cpy), oRBIU_pPIE_susc_cntl_sel(RBIU_PIPE_susc_cntl_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), sr_cntx1to7_busy(sr_cntx1to7_busy), SC_RBBM_cntx17_busy(SC_RBBM_cntx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr),		2 3 4 5 6 7 8 9 10 11 12 13 14 15	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(sc_rs), // read strobe daisy chain out oRBIU_block_rd(sc_rd), // read data daisy chain out // Interface to sc_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_re), oRBIU_re(RBIU_re), oRBIU_waddr(RBIU_waddr), oRBIU_waddr(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata), oRBIU_cpy(RBIU_cpy), oRBIU_pPIPE_susc_cntl_sel(RBIU_PIPE_susc_cntl_sel), oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_misc_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), sr_cntx1to7_busy(sr_cntx1tor_busy), SC_RBBM_entx1T_busy(SC_RBBM_cntx1T_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask));		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(se_rs), // read strobe daisy chain out oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_re), oRBIU_re(RBIU_re), oRBIU_waddr(RBIU_waddr), oRBIU_waddr(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata), oRBIU_cpy(RBIU_cpy), oRBIU_PIPE_susc_cntl_sel(RBIU_PIPE_susc_cntl_sel), oRBIU_PIPE_sq_context_misc(RBIU_PIPE_window_offset_sel), oRBIU_PIPE_window_offset_sel(RBIU_PIPE_window_offset_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(iterator_SP3_valid), sr_entx0_busy(sr_entx0_busy), SC_RBBM_entx0_busy(SC_RBBM_entx0_busy), sr_entx1to7_busy(sr_entx1to7_busy), SC_RBBM_entx17_busy(SC_RBBM_entx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(se_rs), // read strobe daisy chain out oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_re), oRBIU_re(RBIU_re), oRBIU_waddr(RBIU_waddr), oRBIU_waddr(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata), oRBIU_epy(RBIU_cpy), oRBIU_PIPE_susc_cntl_sel(RBIU_PIPE_susc_cntl_sel), oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_mise_sel), oRBIU_PIPE_window_offset_sel(RBIU_PIPE_malconfig_sel), oRBIU_PIPE_aa_config_sel(RBIU_PIPE_aa_config_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), sr_cntx1to7_busy(sr_cntx1to7_busy), SC_RBBM_cntx1T_busy(SC_RBBM_cntx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask));		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_re), .oRBIU_waddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_raddr(RBIU_wadta), .iPIPE_RBIU_rdata(PIPE_RBIU_rdata), .oRBIU_pype_susc_entl_sel(RBIU_PIPE_susc_entl_sel), .oRBIU_PIPE_sa_context_misc(RBIU_PIPE_sq_context_mise_sel), .oRBIU_PIPE_window_offset_sel(RBIU_PIPE_aa_config_sel), .oRBIU_PIPE_aa_config_sel(RBIU_PIPE_aa_config_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_entx0_busy), sr_cntx1to7_busy(sr_cntx1to7_busy), SC_RBBM_cntx1T_busy(SC_RBBM_entx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)); //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain oRBIU_block_rs(se_rs), // read strobe daisy chain out oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe oRBIU_we(RBIU_we), oRBIU_we(RBIU_re), oRBIU_re(RBIU_re), oRBIU_waddr(RBIU_waddr), oRBIU_waddr(RBIU_wdata), iPIPE_RBIU_rdata(PIPE_RBIU_rdata), oRBIU_epy(RBIU_cpy), oRBIU_PIPE_susc_cntl_sel(RBIU_PIPE_susc_cntl_sel), oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_mise_sel), oRBIU_PIPE_window_offset_sel(RBIU_PIPE_malconfig_sel), oRBIU_PIPE_aa_config_sel(RBIU_PIPE_aa_config_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), sr_cntx1to7_busy(sr_cntx1tor_busy), SC_RBBM_entx17_busy(SC_RBBM_cntx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)); //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_re), .oRBIU_waddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_radta(RBIU_wdata), .iPIPE_RBIU_rdata(PIPE_RBIU_rdata), .oRBIU_plPE_susc_entl_sel(RBIU_PIPE_susc_entl_sel), .oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_misc_sel), .oRBIU_PIPE_window_offset_sel(RBIU_PIPE_aa_config_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel), .oRBIU_PIPE_iss_sample_sel_0_sel(RBIU_PIPE_iss_sample_sel_0_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_cntx0_busy), sr_cntx1to7_busy(sr_cntx1tor_busy), SC_RBBM_cntx1T_busy(SC_RBBM_cntx1T_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)); //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_re), .oRBIU_waddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_wadta(RBIU_wadta), .iPIPE_RBIU_rdata(PIPE_RBIU_rdata), .oRBIU_epy(RBIU_cpy), .oRBIU_pIPE_susc_entl_sel(RBIU_PIPE_susc_entl_sel), .oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_mise_sel), .oRBIU_PIPE_window_offset_sel(RBIU_PIPE_aa_enfig_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel), .oRBIU_PIPE_is_s_sample_sel_0_sel(RBIU_PIPE_is_s_sample_sel_0_sel), .oRBIU_PIPE_js_s_sample_sel_0_sel(RBIU_PIPE_js_s_sample_sel_0_sel), .oRBIU_PIPE_js_s_sample_sel_1_sel(RBIU_PIPE_js_s_sample_sel_1_sel), .oRBIU_PIPE_js_s_sample_sel_1_sel(RBIU_PIPE_js_s_sample_sel_1_sel), .oRBIU_PIPE_js_s_sample_sel_1_sel(RBIU_PIPE_js_s_sample_sel_1_sel), .oRBIU_PIPE_js_s_sample_sel_1_sel(RBIU_PIPE_js_s_sample_sel_1_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_last_quad(u3_SC_SP_last_quad), iterator_SP3_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_entx0_busy), sr_cntx1to7_busy(sr_cntx1to7_busy), SC_RBBM_cntx17_busy(SC_RBBM_entx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)); //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 //	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_re), .oRBIU_waddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_wata(RBIU_wadta), .iPIPE_RBIU_rdata(PIPE_RBIU_rdata), .oRBIU_py(RBIU_cpy), .oRBIU_pIPE_susc_entl_sel(RBIU_PIPE_susc_entl_sel), .oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_mise_sel), .oRBIU_PIPE_window_offset_sel(RBIU_PIPE_aa_enfig_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel), .oRBIU_PIPE_iss_sample_sel_0_sel(RBIU_PIPE_iss_sample_sel_0_sel), .oRBIU_PIPE_iss_sample_sel_1_sel(RBIU_PIPE_iss_sample_sel_1_sel), .oRBIU_PIPE_iss_sample_sel_1_sel(RBIU_PIPE_iss_sample_sel_1_sel), .oRBIU_PIPE_iss_sample_sel_1_sel(RBIU_PIPE_iss_sample_sel_1_sel), .oRBIU_PIPE_issa_az_22_offset_sel(RBIU_PIPE_issa_az_22_offset_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_entx0_busy), sr_cntx1to7_busy(sr_cntx1to7_busy), SC_RBBM_cntx17_busy(SC_RBBM_entx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)); //		2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 // 23 //	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_re), .oRBIU_waddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_gy(RBIU_rey), .oRBIU_pPPE_RBIU_rdata(PIPE_RBIU_rdata), .oRBIU_pIPE_susc_entl_sel(RBIU_PIPE_susc_entl_sel), .oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_mise_sel), .oRBIU_PIPE_window_offset_sel(RBIU_PIPE_aa_enfig_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel), .oRBIU_PIPE_iss_sample_sel_0_sel(RBIU_PIPE_iss_sample_sel_0_sel), .oRBIU_PIPE_iss_sample_sel_1_sel(RBIU_PIPE_iss_sample_sel_1_sel), .oRBIU_PIPE_msaa_2x2_offset_sel(RBIU_PIPE_msaa_2x2_offset_sel), .oRBIU_PIPE_msaa_free_tl_0_sel(RBIU_PIPE_msaa_2x2_offset_sel), .oRBIU_PIPE_msaa_free_tl_0_sel(RBIU_PIPE_msaa_2x2_offset_sel), .oRBIU_PIPE_msaa_free_tl_0_sel(RBIU_PIPE_msaa_2x2_offset_sel), .oRBIU_PIPE_aa_offset_tbl_0_sel(RBIU_PIPE_aa_offset_tbl_0_sel),
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	iterator_SP3_data(iterator_SP3_data), u3_SC_SP_data(u3_SC_SP_data), iterator_SP3_type(iterator_SP3_type), u3_SC_SP_type(u3_SC_SP_type), iterator_SP3_last_quad(iterator_SP3_last_quad), u3_SC_SP_last_quad(iterator_SP3_last_quad), iterator_SP3_valid(iterator_SP3_valid), u3_SC_SP_last_quad(u3_SC_SP_last_quad), iterator_SP3_valid(u3_SC_SP_valid), sr_cntx0_busy(sr_cntx0_busy), SC_RBBM_cntx0_busy(SC_RBBM_entx0_busy), sr_cntx1tor_busy(sr_cntx1tor_busy), SC_RBBM_cntx1T_busy(SC_RBBM_entx17_busy), RC_SC_detail_rtr(RC_SC_detail_rtr), rdy_for_detail_mask(rdy_for_detail_mask)); //	Ex. 2118 - sc.y	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 // 23 // 24 //	iRBBM_wd_ql(reg_wd), // write data iRBBM_re_ql(reg_re), // read enable // RBBM read data daisy chain .oRBIU_block_rs(se_rs), // read strobe daisy chain out .oRBIU_block_rd(se_rd), // read data daisy chain out // Interface to se_pipe .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_we), .oRBIU_we(RBIU_red), .oRBIU_maddr(RBIU_waddr), .oRBIU_maddr(RBIU_waddr), .oRBIU_raddr(RBIU_waddr), .oRBIU_gy(RBIU_red), .oRBIU_pPE_RBIU_rdata(PIPE_RBIU_rdata), .oRBIU_pIPE_susc_entl_sel(RBIU_PIPE_susc_entl_sel), .oRBIU_PIPE_sq_context_misc(RBIU_PIPE_sq_context_mise_sel), .oRBIU_PIPE_window_offset_sel(RBIU_PIPE_aa_config_sel), .oRBIU_PIPE_aa_mask_sel(RBIU_PIPE_aa_mask_sel), .oRBIU_PIPE_iss_sample_sel_0_sel(RBIU_PIPE_iss_sample_sel_0_sel), .oRBIU_PIPE_iss_sample_sel_1_sel(RBIU_PIPE_iss_sample_sel_1_sel), .oRBIU_PIPE_msaa_2x2_offset_sel(RBIU_PIPE_msaa_2x2_offset_sel), .oRBIU_PIPE_maa_offset_tbl_0_sel(RBIU_PIPE_aa_offset_tbl_0_sel), .oRBIU_PIPE_aa_offset_tbl_1_sel(RBIU_PIPE_aa_offset_tbl_1_sel), .oRBIU_PIPE_aa_offset_tbl_1_sel(RBIU_PIPE_aa_offset_tbl_1_sel), .oRBIU_PIPE_aa_offset_tbl_1_sel(RBIU_PIPE_aa_offset_tbl_1_sel),

```
.oRBIU PIPE aa offset tbl 3 sel(RBIU PIPE aa offset tbl 3 sel),
                                                                                                                      1 // Instantiate blocks
 2
          .oRBIU PIPE line stipple sel(RBIU PIPE line stipple sel),
          .oRBIU PIPE line stipple state sel(RBIU PIPE line stipple state sel),
          .oRBIU_PIPE_line_cntl_sel(RBIU_PIPE_line_cntl_sel),
                                                                                                                            sc_state usc_state (
          .oRBIU_PIPE_window_scissor_tl_sel(RBIU_PIPE_window_scissor_tl_sel),
                                                                                                                              .iSRST(sc srst),
          .oRBIU_PIPE_window_scissor_br_sel(RBIU_PIPE_window_scissor_br_sel),
                                                                                                                              .iSCLK(sclk_sc),
          .oRBIU_PIPE_screen_scissor_tl_sel(RBIU_PIPE_screen_scissor_tl_sel),
                                                                                                                              .iRBIU wdata(RBIU wdata),
          .oRBIU_PIPE_screen_scissor_br_sel(RBIU_PIPE_screen_scissor_br_sel),
                                                                                                                              .iRBIU_we(RBIU_we),
          .oRBIU_PIPE_cliprect_0_tl_sel(RBIU_PIPE_cliprect_0_tl_sel),
                                                                                                                              .iRBIU re(RBIU re),
10
          .oRBIU\_PIPE\_cliprect\_0\_br\_sel(RBIU\_PIPE\_cliprect\_0\_br\_sel),
                                                                                                                              .iRBIU waddr(RBIU waddr),
11
          .oRBIU_PIPE_cliprect_l_tl_sel(RBIU_PIPE_cliprect_l_tl_sel),
                                                                                                                              .iRBIU_raddr(RBIU_raddr),
12
          .oRBIU_PIPE_cliprect_1_br_sel(RBIU_PIPE_cliprect_1_br_sel),
                                                                                                                              .iRBIU_cpy(RBIU_cpy),
13
                                                                                                                              .iRBIU_PIPE_susc_cntl_sel(RBIU_PIPE_susc_cntl_sel),
          .oRBIU PIPE cliprect 2 tl sel(RBIU PIPE cliprect 2 tl sel),
14
                                                                                                                              .iRBIU_PIPE_sq_context_misc_sel(RBIU_PIPE_sq_context_misc_sel),
          .oRBIU PIPE cliprect 2 br sel(RBIU PIPE cliprect 2 br sel),
15
                                                                                                                    15
                                                                                                                              .iRBIU PIPE window offset sel(RBIU PIPE window offset sel),
          .oRBIU PIPE cliprect 3 tl sel(RBIU PIPE cliprect 3 tl sel),
16
          .oRBIU_PIPE_cliprect_3_br_sel(RBIU_PIPE_cliprect_3_br_sel),
                                                                                                                    16
                                                                                                                              iRBIU PIPE aa config sel(RBIU PIPE aa config sel),
17
                                                                                                                    17
                                                                                                                              .iRBIU PIPE aa mask sel(RBIU PIPE aa mask sel).
          .oRBIU PIPE cliprect rule sel(RBIU PIPE cliprect rule sel).
18
          .oRBIU PIPE poly offset front scale sel(RBIU PIPE poly offset front scale sel).
                                                                                                                    18
                                                                                                                              iRBIU PIPE iss sample sel 0 sel(RBIU PIPE iss sample sel 0 sel),
19
          .oRBIU PIPE poly offset front offset sel(RBIU PIPE poly offset front offset sel),
                                                                                                                    19
                                                                                                                              .iRBIU_PIPE_jss_sample_sel_1_sel(RBIU_PIPE_jss_sample_sel_1_sel),
20
                                                                                                                    20 //
                                                                                                                              .iRBIU PIPE msaa_2x2_offset_sel(RBIU_PIPE_msaa_2x2_offset_sel),
          .oRBIU PIPE poly offset back scale sel(RBIU PIPE poly offset back scale sel),
21
                                                                                                                    21 //
          .oRBIU PIPE poly offset back offset sel(RBIU PIPE poly offset back offset sel),
                                                                                                                              .iRBIU PIPE aa offset tbl 0 sel(RBIU PIPE aa offset tbl 0 sel).
22
          .oRBIU PIPE rt set cntx0 busy(rt set cntx0 busy)
                                                                                                                    22 //
                                                                                                                               .iRBIU PIPE aa offset tbl 1 sel(RBIU PIPE aa offset tbl 1 sel),
23
                                                                                                                    23 //
                                                                                                                               . iRBIU\_PIPE\_aa\_offset\_tbl\_2\_sel(RBIU\_PIPE\_aa\_offset\_tbl\_2\_sel),
24
                                                                                                                    24 //
                                                                                                                               .iRBIU_PIPE_aa_offset_tbl_3_sel(RBIU_PIPE_aa_offset_tbl_3_sel),
                                                                                                                              .iRBIU_PIPE_line_stipple_sel(RBIU_PIPE_line_stipple_sel),
                                          Page 45 of 87
                                                                                                                                                              Page 46 of 87
                                                                             Ex. 2118 - sc.v
                                                                                                                                                                                                  Ex. 2118 - sc.v
          .iRBIU_PIPE_line_stipple_state_sel(RBIU_PIPE_line_stipple_state_sel),
 2
          iRBIU PIPE line cntl sel(RBIU PIPE line cntl sel),
                                                                                                                     2
                                                                                                                              .oPIPE RBIU rdata(PIPE RBIU rdata),
          iRBIU PIPE window scissor tl sel(RBIU PIPE window scissor tl sel),
                                                                                                                              .oST MSAA ENABLE(st msaa enable).
          iRBIU PIPE window scissor br sel(RBIU PIPE window scissor br sel),
                                                                                                                              .oST ITER MSAA_ENABLE(st_iter_msaa_enable),
          .iRBIU_PIPE_screen_scissor_tl_sel(RBIU_PIPE_screen_scissor_tl_sel),
                                                                                                                              .oST AA MASK(st aa mask),
          .iRBIU_PIPE_screen_scissor_br_sel(RBIU_PIPE_screen_scissor_br_sel),
                                                                                                                              .oST_MSAA_NUM_SAMPLES(st_msaa_num_samples),
          . iRBIU\_PIPE\_cliprect\_0\_tl\_sel(RBIU\_PIPE\_cliprect\_0\_tl\_sel),
                                                                                                                              .oST_ITER_MSAA_NUM_SAMPLES(st_iter_msaa_num_samples),
          .iRBIU PIPE cliprect 0 br sel(RBIU PIPE cliprect 0 br sel),
                                                                                                                               .oST SCISSOR EN(st scissor en),
          .iRBIU_PIPE_cliprect_1_tl_sel(RBIU_PIPE_cliprect_1_tl_sel),
                                                                                                                               .oST\_DRAW\_ZERO\_LENGTH\_LINE(st\_draw\_zero\_length\_line),
10
          .iRBIU_PIPE_cliprect_l_br_sel(RBIU_PIPE_cliprect_l_br_sel),
                                                                                                                    10
                                                                                                                              .oST_WINDOW_OFFSET_DISABLE(st_window_offset_disable),
11
          .iRBIU_PIPE_cliprect_2_tl_sel(RBIU_PIPE_cliprect_2_tl_sel),
                                                                                                                    11
                                                                                                                              .oST_WINDOW_SCISSOR_X_MIN(st_window_scissor_x_min),
12
          .iRBIU_PIPE_cliprect_2_br_sel(RBIU_PIPE_cliprect_2_br_sel),
                                                                                                                    12
                                                                                                                              .oST_WINDOW_SCISSOR_X_MAX(st_window_scissor_x_max),
13
          .iRBIU_PIPE_cliprect_3_tl_sel(RBIU_PIPE_cliprect_3_tl_sel),
                                                                                                                              .oST_WINDOW_SCISSOR_Y_MIN(st_window_scissor_y_min),
14
          .iRBIU_PIPE_cliprect_3_br_sel(RBIU_PIPE_cliprect_3_br_sel),
                                                                                                                    14
                                                                                                                              .oST_WINDOW_SCISSOR_Y_MAX(st_window_scissor_y_max),
15
          .iRBIU_PIPE_cliprect_rule_sel(RBIU_PIPE_cliprect_rule_sel),
                                                                                                                              .oST SCREEN SCISSOR X MIN(),
16
          . iRBIU\_PIPE\_poly\_offset\_front\_scale\_sel(RBIU\_PIPE\_poly\_offset\_front\_scale\_sel), \\
                                                                                                                              .oST SCREEN SCISSOR X MAX(),
17
          .iRBIU_PIPE_poly_offset_front_offset_sel(RBIU_PIPE_poly_offset_front_offset_sel),
                                                                                                                              .oST\_SCREEN\_SCISSOR\_Y\_MIN(),
18
          iRBIU PIPE poly offset back scale sel(RBIU PIPE poly offset back scale sel),
                                                                                                                              .oST_SCREEN_SCISSOR_Y_MAX(),
19
          .iRBIU_PIPE_poly_offset_back_offset_sel(RBIU_PIPE_poly_offset_back_offset_sel),
                                                                                                                              .oST_X_OFFSET(st_x_offset),
20
          .iSTATE VAR INDX(sr state var indx),
                                                                                                                    20
                                                                                                                              .oST Y OFFSET(st v offset),
21
          .iSTATE VAR INDX DLY(3'b000),
                                                                                                                    21
                                                                                                                              .oST BRES CNTL EN(st bres cntl en),
22
                                                                                                                    22
          .iZ SV INDX(zff state var indx).
                                                                                                                              .oST BRES CNTL REG(st bres cntl reg),
23
                                                                                                                    23
          .iQPP_SV_INDX(qpp_state_var_indx),
24
                                                                                                                    24
          .iPKR SV INDX(pkr sv indx),
                                                                                                                              .oST OUTPUT SCREEN XY(st output screen xy),
2.5
          .iITER SV INDX(iterator sv indx),
                                                                                                                              .oST LINE STIPPLE ENABLE(st line stipple enable).
                                          Page 47 of 87
                                                                                                                                                              Page 48 of 87
                                                                             Ex. 2118 - sc.v
                                                                                                                                                                                                 Ex. 2118 - sc.v
```

```
.oST\_JSS\_ENABLE(st\_jss\_enable),
                                                                                                                          .oST_MSAA_LRC_SAMP_OFFSET_X(st_msaa_lrc_samp_offset_x),
 2
          .oST_ITER_JSS_ENABLE(st_iter_jss_enable),
                                                                                                                          .oST\_MSAA\_LRC\_SAMP\_OFFSET\_Y(st\_msaa\_lrc\_samp\_offset\_y),
          .oST_JSS_X_DIM(st_jss_x_dim),
                                                                                                                          .oST\_SAMPLE\_0\_X(st\_sample\_0\_x),
          .oST_JSS_Y_DIM(st_jss_y_dim),
                                                                                                                          .oST_SAMPLE_0_Y(st_sample_0_y),
          .oST_MAX_SAMPLE_DIST(st_max_sample_dist),
                                                                                                                          .oST\_SAMPLE\_l\_X(st\_sample\_l\_x),
          .oST_JSS_SAMPLE0_SEL(st_jss_sample0_sel),
                                                                                                                          .oST\_SAMPLE\_l\_Y(st\_sample\_l\_y),
          .oST_JSS_SAMPLE1_SEL(st_jss_sample1_sel),
                                                                                                                          .oST\_SAMPLE\_2\_X(st\_sample\_2\_x),
          .oST_JSS_SAMPLE2_SEL(st_jss_sample2_sel),
                                                                                                                          .oST\_SAMPLE\_2\_Y(st\_sample\_2\_y),
          .oST_JSS_SAMPLE3_SEL(st_jss_sample3_sel),
                                                                                                                         .oST_SAMPLE_3_X(st_sample_3_x),
          .oST_JSS_SAMPLE4_SEL(st_jss_sample4_sel),
                                                                                                                         .oST\_SAMPLE\_3\_Y(st\_sample\_3\_y),
11
          .oST_JSS_SAMPLE5_SEL(st_jss_sample5_sel)
                                                                                                                11
                                                                                                                         .oST_SAMPLE_4_X(st_sample_4_x),
12
          .oST_JSS_SAMPLE6_SEL(st_jss_sample6_sel),
                                                                                                                12
                                                                                                                         .oST_SAMPLE_4_Y(st_sample_4_y),
13
          .oST JSS SAMPLE7 SEL(st jss sample7 sel),
                                                                                                                13
                                                                                                                          .oST SAMPLE 5 X(st sample 5 x),
14
          .oST JSS SAMPLE8 SEL(st jss sample8 sel),
                                                                                                                          .oST SAMPLE 5 Y(st sample 5 y),
15
          .oST JSS SAMPLE9 SEL(st jss sample9 sel),
                                                                                                                15
                                                                                                                          .oST_SAMPLE_6_X(st_sample_6_x),
16
          .oST JSS SAMPLE10 SEL(st jss sample10 sel),
                                                                                                                16
                                                                                                                          .oST_SAMPLE_6_Y(st_sample_6_y),
17
                                                                                                                17
          .oST JSS SAMPLE11 SEL(st jss sample11 sel),
                                                                                                                          .oST_SAMPLE_7_X(st_sample_7_x),
18
          .oST_JSS_SAMPLE12_SEL(st_jss_sample12_sel),
                                                                                                                18
                                                                                                                          .oST_SAMPLE_7_Y(st_sample_7_y),
19
          .oST_JSS_SAMPLE13_SEL(st_jss_sample13_sel),
                                                                                                                19
                                                                                                                          .oST_SAMPLE_8_X(st_sample_8_x),
20
          .oST_JSS_SAMPLE14_SEL(st_jss_sample14_sel),
                                                                                                                20
                                                                                                                          .oST\_SAMPLE\_8\_Y(st\_sample\_8\_y),
21
          .oST_JSS_SAMPLE15_SEL(st_jss_sample15_sel),
                                                                                                                21
                                                                                                                          .oST\_SAMPLE\_9\_X(st\_sample\_9\_x),
22 //
         .oST\_MSAA\_URC\_SAMP\_OFFSET\_X(st\_msaa\_urc\_samp\_offset\_x),
                                                                                                                22
                                                                                                                          .oST\_SAMPLE\_9\_Y(st\_sample\_9\_y),
          .oST\_MSAA\_URC\_SAMP\_OFFSET\_Y(st\_msaa\_urc\_samp\_offset\_y),
                                                                                                                23
                                                                                                                          .oST_SAMPLE_10_X(st_sample_10_x),
          .oST\_MSAA\_LLC\_SAMP\_OFFSET\_X(st\_msaa\_llc\_samp\_offset\_x),
                                                                                                                24
                                                                                                                          .oST_SAMPLE_10_Y(st_sample_10_y),
          .oST\_MSAA\_LLC\_SAMP\_OFFSET\_Y(st\_msaa\_llc\_samp\_offset\_y),
                                                                                                                          .oST_SAMPLE_11_X(st_sample_11_x),
                                                                                                                                                         Page 50 of 87
                                         Page 49 of 87
                                                                          Ex. 2118 - sc.v
                                                                                                                                                                                           Ex. 2118 - sc.v
                                                                                                                          .oST\_CLIPRECT\_2\_X\_MIN(st\_cliprect\_2\_x\_min),
         .oST SAMPLE 11 Y(st sample 11 y),
 2
         .oST SAMPLE_12_X(st_sample_12_x),
                                                                                                                          .oST\_CLIPRECT\_2\_Y\_MIN(st\_cliprect\_2\_y\_min),
          .oST_SAMPLE_12_Y(st_sample_12_y),
                                                                                                                          .oST_CLIPRECT_2_X_MAX(st_cliprect_2_x_max),
          .oST_SAMPLE_13_X(st_sample_13_x),
                                                                                                                          .oST_CLIPRECT_2_Y_MAX(st_cliprect_2_y_max),
          .oST_SAMPLE_13_Y(st_sample_13_y),
                                                                                                                          .oST_CLIPRECT_3_X_MIN(st_cliprect_3_x_min),
          .oST_SAMPLE_14_X(st_sample_14_x),
                                                                                                                          .oST_CLIPRECT_3_Y_MIN(st_cliprect_3_y_min),
          .oST_SAMPLE_14_Y(st_sample_14_y),
                                                                                                                          .oST_CLIPRECT_3_X_MAX(st_cliprect_3_x_max),
          .oST_SAMPLE_15_X(st_sample_15_x),
                                                                                                                          .oST_CLIPRECT_3_Y_MAX(st_cliprect_3_y_max),
          .oST_SAMPLE_15_Y(st_sample_15_y),
                                                                                                                          .oST_CLIP_RULE(st_clip_rule),
10
          .oST_LINE_PATTERN(st_line_pattern),
                                                                                                                10
                                                                                                                          .oST_POLY_OFFSET_PARA_ENABLE(st_poly_offset_para_enable),
11
          .oST_REPEAT_COUNT(st_repeat_count),
                                                                                                                11
                                                                                                                          .oST\_POLY\_OFFSET\_BACK\_ENABLE(st\_poly\_offset\_back\_enable),
12 //
          .oST_PATTERN_START(st_pattern_start),
                                                                                                                12
                                                                                                                          .oST\_POLY\_OFFSET\_FRONT\_ENABLE (st\_poly\_offset\_front\_enable),
13
          .oST_PATTERN_BIT_ORDER(st_pattern_bit_order),
                                                                                                                          .oST_POLY_OFFSET_FRONT_SCALE(st_poly_offset_front_scale),
14
          .oST_AUTO_RESET_ENABLE(st_auto_reset_enable),
                                                                                                                          .oST_POLY_OFFSET_FRONT_OFFSET(st_poly_offset_front_offset),
15
          .oST_CURRENT_PTR(st_current_ptr),
                                                                                                                          .oST_POLY_OFFSET_BACK_SCALE(st_poly_offset_back_scale),
          .oST_CURRENT_COUNT(st_current_count),
                                                                                                                          .oST_POLY_OFFSET_BACK_OFFSET(st_poly_offset_back_offset),
17
                                                                                                                17
          .oST_CLIPRECT_ENABLE(st_cliprect_enable),
                                                                                                                          .oST_SEND_CENTERS(st_send_centers),
18
          .oST_CLIPRECT_0_X_MIN(st_cliprect_0_x_min),
                                                                                                                18
                                                                                                                          .oST SEND CENTROIDS(st send centroids)
19
                                                                                                                19
          .oST_CLIPRECT_0_Y_MIN(st_cliprect_0_y_min),
20
          .oST CLIPRECT_0_X_MAX(st_cliprect_0_x_max),
                                                                                                                20
21
          .oST CLIPRECT_0_Y_MAX(st_cliprect_0_y_max),
                                                                                                                21
                                                                                                                       sc_stage_reg usc_stage_reg(
22
          .oST_CLIPRECT_1_X_MIN(st_cliprect_1_x_min),
                                                                                                                22
                                                                                                                        .iSRST(sc srst).
23
          .oST_CLIPRECT_1_Y_MIN(st_cliprect_1_y_min),
                                                                                                                23
                                                                                                                         .iSCLK(sclk sc).
24
          .oST_CLIPRECT_l_X_MAX(st_cliprect_l_x_max),
                                                                                                                24
                                                                                                                         . iFREEZE\_B (pipe\_freeze\_b\_dly),
2.5
          .oST_CLIPRECT_l_Y_MAX(st_cliprect_l_y_max),
                                                                                                                         .iFREEZE1_B(pipe_freeze_b_dly1),
                                         Page 51 of 87
                                                                                                                                                         Page 52 of 87
                                                                          Ex. 2118 - sc.v
                                                                                                                                                                                          Ex. 2118 - sc.v
```

1	.iVALID(valid_in),	l .oDEAL	LOC_SLOT(sr_dealloc_slot),	
2	.iPHASE(phase_in),		_PRIM_OF_SLOT(sr_first_prim_of_slot),	
3	.iP0(p0_in),		OF_PKT(sr_end_of_pkt),	
4	.iP1(p1_in),		_FACE(sr_back_face),	
5	.iP2(p2_in),		OKING_VERTEX(sr_provoking_vertex),	
6	.iP3(p3_in),		JOR(sr_x_major),	
7	.iP4(p4_in),	_	Γ_IN_DIAMOND(sr_start_in_diamond),	
8	.iXY0(xy0_in),		IN_DIAMOND(sr_end_in_diamond),	
9	.iXY1(xy1_in),		TYPE(sr_prim_type),	
10	.iXY2(xy2_in),	1	E(sr_phase),	
11	.iZMINMAX(zminmax_in),		E_VAR_INDX(sr_state_var_indx),	
12	.iCNTL(cntl_in),	12 .oV0(sr_		
13	.iV0_INDX(v0_indx_in),	13 .oV1(sr_		
14	.iRT_SET_CNTX0_BUSY(rt_set_cntx0_busy),	14 .oV2(sr_		
15	.iCNTX0_DECR(cntx0_decr),		ζ(sr_ref_x),	
16	.iCNTX1TO7_DECR(cntx1to7_decr),	_	/(sr_ref_y),	
17	.iPKR_ITER_CNTX0_BUSY(pkr_iter_cntx0_busy),	17 .oI0(sr_i		
18	.iPKR_ITER_CNTX1TO7_BUSY(pkr_iter_cntx1to7_busy),	18 .oIX(sr_		
19		19 .oIY(sr_		
20	.oPRIM_WE(sr_prim_we),	20 .oJ0(sr_j		
21	.oZ_WE(sr_z_we),	21 .oJX(sr_		
22	.oPIPE_VALID(sr_pipe_valid),	22 .oJY(sr_		
23	.oEVENT(sr_event),	23 .oW0(sr		
24	.oEVENT_ID(sr_event_id),	24 .oWX(sr	_wx),	
25	.oNULL_PRIM(sr_null_prim),	25 .oWY(sr	_wy),	
			Page 54 of 87	
	Page 53 of 87 Ex. 2118 - sc.v			Ex. 2118 - sc.v
	Page 53 of 87 Ex. 2118 - sc.v			Ex. 2118 - sc.v
	Ex. 2118 - se.v			Ex. 2118 - sc.v
1	Ex. 2118 - se.v .oPARAM_CACHE_INDX0(sr_param_cache_indx0),		rim_ff_empty),	Ex. 2118 - sc.v
2	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1),	2 .busy(),	rrim_ff_empty),	Ex. 2118 - sc.v
2	.oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2),	2 .busy(), 3 .write_d.	rrim_ff_empty), ata(prim_ff_wr_data),	Ex. 2118 - sc.v
2 3 4	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff),	2 .busy(), 3 .write_d. 4 .read_da	rrim_ff_empty), sta(prim_ff_wr_data), ta(prim_ff_rd_data),	Ex. 2118 - sc.v
2 3 4 5	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk	rrim_ff_empty), sta(prim_ff_wr_data), ta(prim_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(se,	rrim_ff_empty), sta(prim_ff_wr_data), ta(prim_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc) 7);	rrim_ff_empty), sta(prim_ff_wr_data), ta(prim_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8	rrim_ff_empty), sta(prim_ff_wr_data), ta(prim_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _se), _srst)	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_pull_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPCLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_fer_x zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc, 7); 8 9 10 sc_zfifo u	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_pull_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oREF_Y_ZFF(sr_ref_y_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc, 7); 8 9 10 sc_zfifo u 11 .we(sr_z	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_pull_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_FF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPCLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oREF_Y_ZFF(sr_ref_x_zff), .oREF_Y_ZFF(sr_ref_y_zff), .oZO_ZFF(sr_z0_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), re),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_SFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_polymode_zff), .oPCLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oREF_Y_ZFF(sr_ref_y_zff), .oZO_ZFF(sr_zO_zff), .oZJ_ZFF(sr_zZ_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), re), _full),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPCLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oREF_Y_ZFF(sr_ref_y_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_z0_zff), .oZX_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), re),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_pack_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPCLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oREF_Y_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZX_ZFF(sr_zx_zff), .oZM_ZFF(sr_zx_zff), .oZM_ZFF(sr_zx_zff), .oZM_ZFF(sr_zx_zff), .oZM_N_ZFF(sr_zx_zff), .oZM_N_ZFF(sr_zz_min_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff 13 .full(z_f 14 .empty(z 15 .busy(),	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), re), _full), _ff_empty),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_pack_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPCLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZO_ZFF(sr_z0_zff), .oZX_ZFF(sr_zx_zff), .oZMIN_ZFF(sr_zmin_zff), .oZ_MIN_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmax_zff),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z, 15 .busy(), 16 .write_d	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZX_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff), .oZMIN_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oC_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z, 15 .busy(), 16 .write_d 17 .read_da	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_pack_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZX_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff), .oZMIN_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oC_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy)	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(sclk	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZX_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff), .oZMIN_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oC_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_1 13 .full(z_f,1 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(sclk 19 .reset(sc.	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	.oPARAM_CACHE_INDX0(sr_param_cache_indx0), .oPARAM_CACHE_INDX1(sr_param_cache_indx1), .oPARAM_CACHE_INDX2(sr_param_cache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_pack_face_zff), .oBACK_FACE_ZFF(sr_pack_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZX_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff), .oZMIN_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oC_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy)	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc. 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_1 13 .full(z_f,1 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(sclk 19 .reset(sc. 20);	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_unll_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_Y_ZFF(sr_ref_y_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZV_ZFF(sr_zx_zff), .oZV_ZFF(sr_zmax_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oCMNXO_BUSY(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy));	2 .busy(), 3 .write_d 4 .read_da 5 .clk(sclk 6 .reset(sc, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z) 15 .busy(), 16 .write_d 17 .read_da 18 .clk(sclk 19 .reset(sc, 20); 21	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_unll_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZO_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff), .oZY_ZFF(sr_zx_zff), .oZ_MIN_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy)); sc_primfifo usc_primfifo(2 .busy(), 3 .write_d 4 .read_da 5 .clk(selk 6 .reset(se, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff 13 .full(z_f, 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(selk 19 .reset(se, 20); 21 22	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc), _srst)	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_ull_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_zz_zff), .oZV_ZFF(sr_zx_zff), .oZV_ZFF(sr_zx_zff), .oZ_MIN_ZFF(sr_zmax_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy)); sc_primfifo use_primfifo(.we(sr_prim_we),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(selk 6 .reset(se, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(selk 19 .reset(se, 20); 21 22 23 assign st_	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _sc), _srst) sc_zfifo(_we), _re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc),	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_unll_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZV_ZFF(sr_zx_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy)); sc_primfifo use_primfifo(.we(sr_prim_we), .re(prim_ff_re),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(selk 6 .reset(se, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff) 13 .full(z_f, 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(selk 19 .reset(se, 20); 21 22 23 assign st_ 24	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), sc), srst) sc_zfifo(_we), re), f_full), ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc), srst) scissor_en = l'b1;	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_null_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ef_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZV_ZFF(sr_zx_zff), .oZMAX_ZFF(sr_z_min_zff), .oZ_MAX_ZFF(sr_zmin_zff), .oZ_MAX_ZFF(sr_cmx1to7_busy)); se_primfifo use_primfifo(.we(sr_prim_we), .re(prim_ff_re), .full(prim_ff_full),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(selk 6 .reset(se, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff_ 13 .full(z_f, 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(selk 19 .reset(se, 20); 21 22 23 assign st_	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), _se), _srst) se_zfifo(_we), re), _full), _ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _se), _srst) seissor_en = 1'b1;	Ex. 2118 - sc.v
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.oPARAM_CACHE_INDX0(sr_param_eache_indx0), .oPARAM_CACHE_INDX1(sr_param_eache_indx1), .oPARAM_CACHE_INDX2(sr_param_eache_indx2), .oNULL_PRIM_ZFF(sr_unll_prim_zff), .oZY_MAX_ZFF(sr_zy_max_zff), .oBACK_FACE_ZFF(sr_back_face_zff), .oPRIM_TYPE_ZFF(sr_prim_type_zff), .oPOLYMODE_ZFF(sr_polymode_zff), .oSTATE_VAR_INDX_ZFF(sr_state_var_indx_zff), .oREF_X_ZFF(sr_ref_x_zff), .oZO_ZFF(sr_z0_zff), .oZV_ZFF(sr_zx_zff), .oZV_ZFF(sr_zx_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oZ_MAX_ZFF(sr_zmax_zff), .oCNTX0_BUSY(sr_cntx0_busy), .oCNTX1TO7_BUSY(sr_cntx1to7_busy)); sc_primfifo use_primfifo(.we(sr_prim_we), .re(prim_ff_re),	2 .busy(), 3 .write_d 4 .read_da 5 .clk(selk 6 .reset(se, 7); 8 9 10 sc_zfifo u 11 .we(sr_z 12 .re(z_ff) 13 .full(z_f, 14 .empty(z 15 .busy(), 16 .write_d 17 .read_da 18 .clk(selk 19 .reset(se, 20); 21 22 23 assign st_ 24	rrim_ff_empty), ata(prim_ff_wr_data), ta(prim_ff_rd_data), sc), srst) sc_zfifo(_we), re), f_full), ff_empty), ata(z_ff_wr_data), ta(z_ff_rd_data), _sc), srst) scissor_en = l'b1;	Ex. 2118 - sc.v

2 iSCLK(sclk_se), 2 iST_3 3 iSRST(sc_srst), 3 iST_4 4 iPIPE_RTS(sr_pipe_valid), 4 iST_5 5 iPIPE_PHASE(sr_phase), 5 iST_6 6 iEVENT_iD(sr_event), 6 iST_7 7 iEVENT_ID(sr_event_id), 7 8 iX_MAJOR(sr_x_major), 8 oPIP 9 iSTART_IN_DIAMOND(sr_start_in_diamond), 10 oEV 10 iEND_IN_DIAMOND(sr_end_in_diamond), 10 oEV 11 iNULL_PRIM(sr_null_prim), 11 oEV 12 iPRIM_TYPE(sr_prim_type), 12 oE0 13 iV0(sr_v0), 13 oE0 14 iV1(sr_v1), 14 oE0 15 iV2(sr_v2), 15 oE0 16 iV1_EARLY(xy0_in), 17 oE1 17 iV1_EARLY(xy1_in), 17 oE1 18 iST_MSA_ENABLE(st_insaa_enable), 19 oE1 20 iST_MSA_SAMPLE_DIST(st_max_sample_dist), 20 oE2 21	Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
2 iSCLK(selk sc), 2 iST 3 iSRST(sc_srst), 3 iST 4 iPIPE_RTS(sr_pipe_valid), 4 iST 5 iPIPE_PHASE(sr_phase), 5 iST 6 iEVENT(sr_event), 6 iST 7 iEVENT_ID(sr_event_id), 7 8 iX_MAJOR(sr_x_major), 8 oPIP 9 iSTART_IN_DIAMOND(sr_start_in_diamond), 9 oPIP 10 iEND_IN_DIAMOND(sr_end_in_diamond), 10 oEV 11 iNULL_PRIM(sr_null_prim), 11 oEV 12 iPRIM_TYPE(sr_prim_type), 12 oE0 13 iV0(sr_v0), 13 oE0 14 iV1(sr_v1), 14 oE0 15 iV2(sr_v2), 15 oE0 16 iV2(sr_v2), 15 oE0 17 iV1_EARLY(xy0_in), 17 oE1 18 iST_MSA_ENABLE(st_iss_enable), 19 oE1 20 iST_MSA_ENABLE(st_iss_enable), 20 oE2 21 iST_MAX_SAMPLE_DIST(st_max	Y_MAX(st_window_scissor_y_max), X_OFFSET(st_x_offset), Y_OFFSET(st_y_offset), BRES_CNTL_EN(st_bres_cntl_en), BRES_CNTL_REG(st_bres_cntl_reg), E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
2 .iSCLK(selk_se), 2 .iST_3 3 .iSRST(se_srst), 3 .iSRST(se_srst), 3 .iSRST(se_srst), 4 .iPPE_RTS(sr_pipe_valid), 4 .iST_5 5 .iPPE_PHASE(sr_phase), 5 .iST_6 6 .iEVENT(sr_event), 6 .iST_7 7 .iEVENT_ID(sr_event_id), 7 .iEVENT_ID(sr_event_id), 8 .iX_MAJOR(sr_x_major), 8 .oPIP 9 .iSTART_IN_DIAMOND(sr_start_in_diamond), 9 .oPIP 10 .iEND_IN_DIAMOND(sr_end_in_diamond), 10 .oEV 11 .iNULL_PRIM(sr_null_prim), 11 .oEV 12 .iPRIM_TYPE(sr_prim_type), 12 .oE0(14 .iV1(sr_v1), 14 .oE0_14 .iV1(sr_v1), 15 .iV2(sr_v2), 16 .iV2(sr_v2), 16 .iV2(sr_v2), 16 .iV1_EARLY(xy0_in), 17 .oE1_15 .iV1_EARLY(xy1_in), 17 .oE1_16 .iV1_EARLY(xy1_in), 18 .iST_MSAA_ENABLE(st_insae_enable), 19 .oE1_17 .iV1_EARLY(xy1_in), 19 .oE1_18 .iST_MSAA_ENABLE(st_inse_tipple_enable), 19 .oE1_18 .iST_INE_STIPPLE_ENABLE(st_ine_stipple_enable), 19 .oE2_2 .iST_SCISSOR_EN(st_scissor_en), 10 .oE2_24 .iST_AMX_SAMPLE_DIST(st_draw_zero_length_line), 10 .oE2_25 .oX_inc_start_in_	Y_MAX(st_window_scissor_y_max), X_OFFSET(st_x_offset), Y_OFFSET(st_y_offset), BRES_CNTL_EN(st_bres_cntl_en), BRES_CNTL_REG(st_bres_cntl_reg), E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
3 iSRST(sc srst), 4 iPIPE_RTS(sr_pipe_valid), 5 iPIPE_PHASE(sr_phase), 6 iEVENT(sr_event), 7 iEVENT_ID(sr_event_id), 8 iX_MAJOR(sr_x_major), 9 iSTART_IN_DIAMOND(sr_start_in_diamond), 10 iEND_IN_DIAMOND(sr_end_in_diamond), 11 iNULL_PRIM(sr_null_prim), 12 iPRIM_TYPE(sr_prim_type), 13 iV0(sr_v0), 14 iV1(sr_v1), 15 iV2(sr_v2), 16 iV0_EARLY(xy0_in), 17 iST_MSA_ENABLE(st_msaa_enable), 18 iST_MSA_ENABLE(st_msaa_enable), 19 iST_MSA_SAMPLE_DIST(st_max_sample_dist), 20 iST_MAX_SAMPLE_DIST(st_max_sample_dist), 21 iST_LNE_STIPPLE_ENABLE(st_line_stipple_enable), 22 iST_SCISSOR_EN(st_scissor_en), 23 iST_NAA_ZERO_LENGTH_LINE(st_draw_zero_length_line), 24 iST_X_MAX(st_window_scissor_x_max), 25 iST_X_MAX(st_window_scissor_x_max), 26 iST_X_MAX(st_window_scissor_x_max), 27 iST_X_MAX(st_window_scissor_x_max), 28 iST_X_MAX(st_window_scissor_x_max), 29 iST_X_MAX(st_window_scissor_x_max), 20 iST_X_MAX(st_window_scissor_x_max), 20 iST_X_MAX(st_window_scissor_x_max), 20 iST_X_MAX(st_window_scissor_x_max), 21 iST_X_MAX(st_window_scissor_x_max), 22 iST_X_MAX(st_window_scissor_x_max),	X_OFFSET(st_x_offset), Y_OFFSET(st_y_offset), BRES_CNTL_EN(st_bres_cntl_en), BRES_CNTL_REG(st_bres_cntl_reg), E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
4 iSPIE_RTS(sr_pipe_valid), 4 iST_5 iPIE_PHASE(sr_phase), 5 iST_6 iEVENT(sr_event), 6 iSVENT(sr_event), 6 iSVENT(sr_event), 7 iEVENT_ID(sr_event_id), 7 iEVENT_ID(sr_event_id), 8 iX_MAJOR(sr_x_major), 8 oPIE_9 iSTART_IN_DIAMOND(sr_start_in_diamond), 9 oPIE_10 i.END_IN_DIAMOND(sr_end_in_diamond), 10 oEV_11 i.NULL_PRIM(sr_null_prim), 11 oEV_12 i.PRIM_TYPE(sr_prim_type), 12 oE0(13 i.V0(sr_v0), 13 oE0_14 i.V1(sr_v1), 14 oE0_15 i.V2(sr_v2), 15 oE0_16 i.V0_EARLY(xy0_in), 16 oE1(14 i.V1_EARLY(xy1_in), 17 oE1_16 i.ST_MSA_ENABLE(st_insa_enable), 18 oE1_19 i.ST_MSA_ENABLE(st_insa_enable), 19 oE1_10 i.ST_MSA_ENABLE(st_insa_enable),	Y_OFFSET(st_y_offset), BRES_CNTL_EN(st_bres_cntl_en), BRES_CNTL_REG(st_bres_cntl_reg), E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
5 .iPIPE_PHASE(sr_phase), 5 .iST_6 6 .iEVENT(sr_event), 6 .iST_7 7 .iEVENT_ID(sr_event_id), 7 8 .iX_MAJOR(sr_x_major), 8 .oPIP 9 .iSTART_IN_DIAMOND(sr_start_in_diamond), 9 .oPIP 10 .iEND_IN_DIAMOND(sr_end_in_diamond), 10 .oEV 11 .iNULL_PRIM(sr_null_prim), 11 .oEV 12 .iPRIM_TYPE(sr_prim_type), 12 .oE0 13 .iV0(sr_v0), 13 .oE0 14 .iV1(sr_v1), 14 .oE0 15 .iV2(sr_v2),	BRES_CNTL_EN(st_bres_cntl_en), BRES_CNTL_REG(st_bres_cntl_reg), E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
6 iEVENT(sr_event), 7 iEVENT_ID(sr_event_id), 8 iX_MAJOR(sr_x_major), 9 iSTART_IN_DIAMOND(sr_start_in_diamond), 10 iEVEND_IN_DIAMOND(sr_end_in_diamond), 11 iNULL_PRIM(sr_null_prim), 12 iPRIM_TYPE(sr_prim_type), 13 iV(sr_v0), 14 iV1(sr_v1), 15 iV2(sr_v2), 16 iV0_EARLY(xy0_in), 17 iV1_EARLY(xy1_in), 18 iST_MSA_ENABLE(st_ins_enable), 19 iST_JSS_ENABLE(st_iss_enable), 20 iST_MAX_SAMPLE_DIST(st_max_sample_dist), 21 iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 22 iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 24 iST_X_MAX(st_window_scissor_x_max), 26 po PI	BRES_CNTL_REG(st_bres_cntl_reg), E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
7 .iEVENT_ID(sr_event_id), 7 8 .iX_MAJOR(sr_x_major), 8 .oPIP 9 .iSTART_IN_DIAMOND(sr_start_in_diamond), 9 .oPIP 10 .iEND_IN_DIAMOND(sr_end_in_diamond), 10 .oEV 11 .iNULL_PRIM(sr_null_prim), 11 .oEV 12 .iPRIM_TYPE(sr_prim_type), 12 .oE0 13 .iV0(sr_v0), 13 .oE0 14 .iV1(sr_v1), 14 .oE0 15 .iV2(sr_v2), 15 .oE0 16 .iV0_EARLY(xy0_in), 16 .oE1(17 .iV1_EARLY(xy1_in), 17 .oE1 18 .oE1 .iST_MSA_ENABLE(st_msa_enable), 18 .oE1 19 .iST_JSS_ENABLE(st_jss_enable), 20 .oE2 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 22 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 23 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_i <td< td=""><td>E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),</td></td<>	E_RTS(pipe_rts), E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
8 iX_MAJOR(sr_x_major), 8 oPIP 9 iSTART_IN_DIAMOND(sr_start_in_diamond), 9 oPIP 10 iEND_IN_DIAMOND(sr_end_in_diamond), 10 oEV 11 iNULL_PRIM(sr_null_prim), 11 oEV 12 iPRIM_TYPE(sr_prim_type), 12 oE0 13 iV0(sr_v0), 13 oE0 14 iV1(sr_v1), 14 oE0 15 iV2(sr_v2), 15 oE0 16 iV0_EARLY(xy0_in), 16 oE1(17 iV1_EARLY(xy1_in), 17 oE1 18 oE1_ iST_MSA_ENABLE(st_iss_enable), 18 oE1 20 iST_MS_ENBLE(st_iss_enable), 19 oE1 20 iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 oE2 21 iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 oE2 22 iST_SCISSOR_EN(st_scissor_en), 22 oE2 23 iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 oE2 24 iST_X_MIN(st_window_scissor_x_min), 24 oX_i <tr< td=""><td>E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),</td></tr<>	E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
9 .oPIF 10 .isTART_IN_DIAMOND(sr_start_in_diamond), 11 .inULL_PRIM(sr_null_prim), 12 .iPRIM_TYPE(sr_prim_type), 13 .iV0(sr_v0), 14 .iV1(sr_v1), 15 .iV2(sr_v2), 16 .iV2(sr_v2), 17 .iV1_EARLY(xy0_in), 18 .iST_MSA_ENABLE(st_ins_atable), 19 .iST_MSA_ENABLE(st_ins_atable), 19 .iST_MSA_SAMPLE_DIST(st_max_sample_dist), 20 .iST_MAX_SAMPLE_DIST(st_ins_tipple_enable), 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 22 .iST_SCISSOR_EN(st_scissor_en), 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 24 .iST_X_MAX(st_window_scissor_x_min), 25 .iST_X_MAX(st_window_scissor_x_max), 26 .oSJ	E_PHASE(pipe_phase), ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
10 iEND_IN_DIAMOND(sr_end_in_diamond), 10 oEV 11 iNULL_PRIM(sr_null_prim), 11 oEV 12 iPRIM_TYPE(sr_prim_type), 12 oE0(13 o.6D(13 o.6D(14 i.V1(sr_v1), 14 o.6D(15 i.V2(sr_v2), 15 o.6D(16 i.V0_EARLY(xy0_in), 16 o.6D(17 o.V1_EARLY(xy1_in), 17 o.6L(18 o.6L(o.6L(o.6L(19 i.ST_MSA_ENABLE(st_msaa_enable), 18 o.6L(20 i.ST_JSS_ENABLE(st_jss_enable), 20 o.6L(21 i.ST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 o.6L(21 i.ST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 o.6L(22 i.ST_SCISOR_EN(st_scissor_en), 22 o.6L(23 i.ST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 o.6L(24 i.ST_X_MIN(st_window_scissor_x_min), 24 o.X_i 25 o.X_i	ENT(event_flag), ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
11 .iNULL_PRIM(sr_null_prim), 11 .oEV 12 .iPRIM_TYPE(sr_prim_type), 12 .oE0 13 .iV0(sr_v0), 13 .oE0 14 .iV1(sr_v1), 14 .oE0 15 .iV2(sr_v2), 15 .oE0 16 .iV0_EARLY(xy0_in), 16 .oE1(17 .iV1_EARLY(xy1_in), 17 .oE1 18 .iST_MSA_ENABLE(st_msaa_enable), 18 .oE1 19 .iST_JSS_ENABLE(st_jss_enable), 19 .oE1 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_i 25 .oX_i	ENT_ID(event_id), e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
12 .iPRIM_TYPE(sr_prim_type), 12 .oE0(13 .iV0(sr_v0), 13 .oE0 14 .iV1(sr_v1), 14 .oE0 15 .iV2(sr_v2), 15 .oE0 16 .iV0_EARLY(xy0_in), 16 .oE1(17 .iV1_EARLY(xy1_in), 17 .oE1 18 .oE1 .oE1 19 .iST_JSS_ENABLE(st_issa_enable), 19 .oE1 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2(21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_j 25 .oX_j	e0), Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
13 iV0(sr_v0), 13 oE0 14 iV1(sr_v1), 14 oE0 15 iV2(sr_v2), 15 oE0 16 iV0_EARLY(xy0_in), 16 oE1 17 iV1_EARLY(xy1_in), 17 oE1 18 oE1_ isT_MSA_ENABLE(st_iss_enable), 19 oE1 20 iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 oE2 21 iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 oE2 22 iST_SCISSOR_EN(st_scissor_en), 22 oE2 23 iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 oE2 24 iST_X_MIN(st_window_scissor_x_min), 24 oX_i 25 oX_i	Y(e0_y), DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
14 .iVI(sr_v1), 14 .oE0 15 .iV2(sr_v2), 15 .oE0 16 .iV0_EARLY(xy0_in), 16 .oE1 17 .iV1_EARLY(xy1_in), 17 .oE1 18 .iST_MSA_ENABLE(st_msaa_enable), 18 .oE1 19 .iST_JSS_ENABLE(st_jss_enable), 20 .oE2 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_j 25 .oX_j Page 57 of 87	DX(e0_dx), DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
15 i.V2(sr_v2), 15 oE0 16 i.V0_EARLY(xy0_in), 16 oE1 17 i.V1_EARLY(xy1_in), 17 oE1 18 i.ST_MSAA_ENABLE(st_msaa_enable), 18 oE1 19 i.ST_JSS_ENABLE(st_jss_enable), 19 oE1 20 i.ST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 oE2 21 i.ST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 oE2 22 i.ST_SCISSOR_EN(st_scissor_en), 22 oE2 23 i.ST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 oE2 24 i.ST_X_MIN(st_window_scissor_x_min), 24 oX_j 25 o.X_j	DY(e0_dy), e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
16 iV0_EARLY(xy0_in), 16 oE1 17 iV1_EARLY(xy1_in), 17 oE1 18 .iST_MSAA_ENABLE(st_msaa_enable), 18 .oE1 19 .iST_JSS_ENABLE(st_jss_enable), 19 .oE1 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_j 25 .oX_j	e1), X(e1_x), DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
17 .iVI_EARLY(xyl_in), 17 .oE1 18 .iST_MSAA_ENABLE(st_msaa_enable), 18 .oE1 19 .iST_JSS_ENABLE(st_jss_enable), 19 .oE1 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_j 25 .oX_j Page 57 of 87	$X(e1_x)$, $DX(e1_dx)$, $DY(e1_dy)$, $e2$), $X(e2_x)$, $DX(e2_dx)$,
18 .iST_MSAA_ENABLE(st_msaa_enable), 18 oE1 19 .iST_JSS_ENABLE(st_jss_enable), 19 oE1 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_J 25 .oX_J Page 57 of 87	DX(e1_dx), DY(e1_dy), e2), X(e2_x), DX(e2_dx),
19 .iST_JSS_ENABLE(st_jss_enable), 19 oE1 20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_j 25 .oX_j Page 57 of 87	DY(e1_dy), e2), X(e2_x), DX(e2_dx),
20 .iST_MAX_SAMPLE_DIST(st_max_sample_dist), 20 .oE2 21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_J 25 .iST_X_MAX(st_window_scissor_x_max), 25 .oX_J	e2), X(e2_x), DX(e2_dx),
21 .iST_LINE_STIPPLE_ENABLE(st_line_stipple_enable), 21 .oE2 22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ZERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_J 25 .iST_X_MAX(st_window_scissor_x_max), 25 .oX_J	X(e2_x), DX(e2_dx),
22 .iST_SCISSOR_EN(st_scissor_en), 22 .oE2 23 .iST_DRAW_ERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_1 25 .iST_X_MAX(st_window_scissor_x_max), 25 .oX_1	DX(e2_dx),
23 .iST_DRAW_ERO_LENGTH_LINE(st_draw_zero_length_line), 23 .oE2 24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_1 25 .iST_X_MAX(st_window_scissor_x_max), 25 .oX_1 Page 57 of 87	
24 .iST_X_MIN(st_window_scissor_x_min), 24 .oX_ 25 .iST_X_MAX(st_window_scissor_x_max), 25 .oX_ Page 57 of 87	DV(e2 dv)
25 .iST_X_MAX(st_window_scissor_x_max),	DY(e2_dy),
Page 57 of 87	DIR(x_dir),
Page 57 of 87 Ex. 2118 - sc.v	START(x_start),
Ex. 2118 - sc.v	Page 58 of 87
	Ex. 2118 - sc.v
1 .oX_END(x_end), l .iE1_	$DY(e1_dy),$
2 .oY_DIR(y_dir), 2 .iE2(:2),
3 .oY_START(y_start), 3 .iE2_	$X(e2_x),$
4 .oY_END(y_end), 4 .iE2_	$DX(e2_dx),$
5 .oBB_FRACT_BITS(bb_fract_bits), 5 .iE2_	DY(e2_dy),
	DIR(x_dir),
	TART(x_start),
	ND(x_end),
	DIR(y_dir),
	TART(y_start),
	ND(y_end),
12 .iQM 12 .iQM	ASK_FF_ALM_FULL(tile_ff_full),
	RTR(rc_rtr),
	· · · · · · · · ·
14 .iPIPE_PHASE(pipe_phase), 14	
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV	ENT(cw_event),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV	ENT(cw_event), ENT_ID(cw_event_id),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB	ENT(cw_event), ENT_ID(cw_event_id), _FRACT_BITS(cw_bb_fract_bits),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA	ENT(cw_event), ENT_ID(cw_event_id), _FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(e0), 19 .oE0(e)	ENT(cw_event), ENT_ID(cw_event_id), _FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(e0), 19 .oE0 20 .iE0_Y(e0_y), 20 .oE0	ENT(cw_event), ENT_ID(cw_event_id), FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0), Y(cw_e0y),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(e0), 19 .oE0 20 .iE0_Y(e0_y), 20 .oE0 21 .iE0_DX(e0_dx), 21 .oE0	ENT(cw_event), ENT_ID(cw_event_id), FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0), Y(cw_e0y), DX(cw_dxe0),
14 iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(e0), 19 .oE0 20 .iE0_Y(e0_y), 20 .oE0 21 .iE0_DX(e0_dx), 21 .oE0 22 .iE0_DY(e0_dy), 22 .oE0	ENT(cw_event), ENT_ID(cw_event_id), FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0), Y(cw_e0y), DX(cw_dxe0), DY(cw_dye0),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(o_y), 20 .oE0 20 .iE0_Y(e0_y), 20 .oE0 21 .iE0_DX(e0_dx), 21 .oE0 22 .iE0_DY(e0_dy), 22 .oE0 23 .iE1(e1), 23 .oE1(e1)	ENT(cw_event), ENT_ID(cw_event_id), FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0), Y(cw_e0y), DX(cw_dxe0), DY(cw_dye0), cw_e1),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(0), 20 .oE0 20 .iE0_Y(e0_y), 20 .oE0 21 .iE0_DX(e0_dx), 21 .oE0 22 .iE0_DY(e0_dy), 22 .oE0 23 .iE1(e1), 23 .oE1(24 .iE1_X(e1_x), 24 .oE1	ENT(cw_event), ENT_ID(cw_event_id), FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0), Y(cw_e0), DX(cw_dxe0), DY(cw_dve0), cw_e1), X(cw_e1), X(cw_e1x),
14 .iPIPE_PHASE(pipe_phase), 14 15 .iEVENT(event_flag), 15 .oEV 16 .iEVENT_ID(event_id), 16 .oEV 17 .iBB_FRACT_BITS(bb_fract_bits), 17 .oBB 18 .iPASS_EMPTY_PRIM(pass_empty_prim), 18 .oPA 19 .iE0(e0), 19 .oE0 20 .iE0_Y(e0_y), 20 .oE0 21 .iE0_DX(e0_dx), 21 .oE0 22 .iE0_DY(e0_dy), 22 .oE0 23 .iE1(e1), 23 .oE1(24 .iE1_X(e1_x), 24 .oE1	ENT(cw_event), ENT_ID(cw_event_id), FRACT_BITS(cw_bb_fract_bits), SS_EMPTY_PRIM(cw_pass_empty_prim), cw_e0), Y(cw_e0y), DX(cw_dxe0), DY(cw_dye0), cw_e1),

1	.oE1_DY(cw_dye1),		1 sc_quadmask usc_quadmask(
2	.oE2(cw_e2),		2 .iSCLK(sclk_sc),
3	.oE2_X(cw_e2x),		3 .freezeb(qmsk_z_freeze_b),
4	.oE2_DX(cw_dxe2),		4 .tile_valid(cw_tile_valid),
5	.oE2_DY(cw_dye2),		5 .event_in(cw_event),
6	.oX_DIR(cw_xdir),		6 .event_id_in(cw_event_id),
7	.oX_CURR(cw_tilex),		7 .last_tile_in(cw_last_tile),
8	.oX_MIN(cw_xmin),		8 .pass_empty_prim_in(cw_pass_empty_prim),
9	.oX_MAX(cw_xmax),		9 .xdir_in(cw_xdir),
10	.oY_DIR(cw_ydir),		10 .ydir_in(cw_ydir),
11	.oY_CURR(cw_tiley),		11 .xmajor_in(cw_xmajor),
12	.oY_MIN(cw_ymin),		12 .tilex_in(cw_tilex),
13	.oY_MAX(cw_ymax),		13 .tiley_in(cw_tiley),
14	.oLAST_TILE_OF_PRIM(cw_last_tile),		14 .xmin_in(cw_xmin),
15	.oZ_FF_RD_EN(z_ff_re),		15 .xmax_in(cw_xmax),
16	.oPRIM_RTS(cw_tile_valid),		16 .ymin_in(cw_ymin),
17	.oPIPE_FREEZE_B_EARLY(pipe_freeze_b_early),		17 .ymax_in(cw_ymax),
18	.oPIPE_FREEZE_B_DLY(pipe_freeze_b_dly),		18 .bb_fract_bits_in(cw_bb_fract_bits),
19	.oPIPE_FREEZE_B_DLY1(pipe_freeze_b_dly1),		19 .e0y(cw_e0y),
20	.oQMSK_Z_FREEZE_B(qmsk_z_freeze_b)		20 .elx(cw_elx),
21 22);		21 .e2x(cw_e2x), 22 .dxe0 in(cw dxe0),
	// tong daises soutil size and daises		
23	// rwr - tmp drivers until sigs are driven		23
24	assign cw_xmajor = 1'b0;		24
25			2.5 .dye1_m(ew_dye1),
25			
25	Page 61 of 87	Fx 2118 - sc v	Page 62 of 87
25	Page 61 of 87	Ex. 2118 - sc.v	Page 62 of 87 Ex. 2118 - sc.v
25		Ex. 2118 - sc.v	Ex. 2118 - sc.v
	Page 61 of 87 .dxe2_in(cw_dxe2), .dye2_in(cw_dye2),	Ex. 2118 - sc.v	Page 62 of 87 Ex. 2118 - sc.v 1
1	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1 .dxe0_out(qm_dxe0),
1 2	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2), .e0_in(ew_e0),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1
1 2 3	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1
1 2 3 4	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .el_in(cw_e1),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1
1 2 3 4 5	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .el_in(cw_e1),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1
1 2 3 4 5 6	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1
1 2 3 4 5 6	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile),	Ex. 2118 - sc.v	Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 9 10	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 6 7 8 9 10 11 12 13	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(ew_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_ydir),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tiley),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_ydir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tilex_m3_out(qm_tilex_m3),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_ydir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley), .tilex_m3_out(qm_tiley_m3), .tiley_m3_out(qm_tiley_m3),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_ydir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley), .tilex_m3_out(qm_tilex_m3), .tiley_m3_out(qm_tiley_m3), .xmin_out(qm_xmin),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_ydir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley), .tilex_m3_out(qm_tiley_m3), .xmin_out(qm_xmin), .xmax_out(qm_xmin), .xmax_out(qm_xmax),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .el_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley), .tiley_m3_out(qm_tiley_m3), .xmin_out(qm_xmin), .xmax_out(qm_xmin), .xmax_out(qm_xmax), .ymin_out(qm_xmin),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tilex), .tiley_m3_out(qm_tiley_m3), .xmin_out(qm_xmin), .xmax_out(qm_xmax), .ymin_out(qm_ymin), .ymax_out(qm_ymin), .ymax_out(qm_ymax),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2), .e0_in(ew_e0), .el_in(ew_e1), .e2_in(ew_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley), .tiley_m3_out(qm_tiley_m3), .xmin_out(qm_xmin), .xmax_out(qm_xmax), .ymin_out(qm_ymin), .ymax_out(qm_ymax), .bb_fract_bits_out(qm_bb_fract_bits),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	.dxc2_in(cw_dxc2), .dyc2_in(cw_dyc2), .e0_in(cw_ed), .e1_in(cw_el), .e1_in(cw_el), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event_id), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley, .tilex_m3_out(qm_tilex_m3), .xmin_out(qm_xmin), .xmax_out(qm_xmax), .ymin_out(qm_ymin), .ymax_out(qm_ymin), .ymax_out(qm_ymax), .bb_fract_bits_out(qm_bb_fract_bits), .z_mask_needed_out(qm_z_mask_needed),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2), .e0_in(ew_e0), .el_in(ew_e1), .e2_in(ew_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event)d), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tiley), .tilex_m3_out(qm_tiley_m3), .xmin_out(qm_xmin), .xmax_out(qm_xmin), .xmax_out(qm_xmin), .xmax_out(qm_ymin), .ymax_out(qm_ymin), .ymax_out(qm_ymin), .ymax_out(qm_ymax), .bb_fract_bits_out(qm_bb_fract_bits), .z_mask_needed_out(qm_z_mask_needed), .e0_out(qm_e0),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2), .e0_in(ew_e0), .el_in(ew_e1), .e2_in(ew_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event)d), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tiley), .tilex_m3_out(qm_tiley_m3), .xmin_out(qm_xmai), .xmax_out(qm_xmax), .ymin_out(qm_ymax), .bb_fract_bits_out(qm_bb_fract_bits), .z_mask_needed_out(qm_z_mask_needed), .e0_out(qm_e0), .el_out(qm_e1),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.dxe2_in(cw_dxe2), .dye2_in(cw_dye2), .e0_in(cw_e0), .e1_in(cw_e1), .e2_in(cw_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event), .event_id_out(qm_event), .xmajor_out(qm_xdir), .xmajor_out(qm_manjor), .tilex_out(qm_tilex), .tiley_out(qm_tilex), .tiley_out(qm_tiley, m3), .xmin_out(qm_xmin), .xmax_out(qm_xmax), .ymin_out(qm_ymin), .ymax_out(qm_ymin), .ymax_out(qm_ymax), .bb_fract_bits_out(qm_bb_fract_bits), .z_mask_needed_out(qm_z_mask_needed), .e0_out(qm_e0), .e1_out(qm_e2),		Ex. 2118 - sc.v 1
1 2 3 4 5 6 7 8 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	.dxe2_in(ew_dxe2), .dye2_in(ew_dye2), .e0_in(ew_e0), .el_in(ew_e1), .e2_in(ew_e2), .last_tile_out(qm_last_tile), .event_out(qm_event), .event_id_out(qm_event)d), .xdir_out(qm_xdir), .ydir_out(qm_xdir), .xmajor_out(qm_xmajor), .tilex_out(qm_tilex), .tiley_out(qm_tiley), .tilex_m3_out(qm_tiley_m3), .xmin_out(qm_xmai), .xmax_out(qm_xmax), .ymin_out(qm_ymax), .bb_fract_bits_out(qm_bb_fract_bits), .z_mask_needed_out(qm_z_mask_needed), .e0_out(qm_e0), .el_out(qm_e1),		Ex. 2118 - sc.v 1

```
.oZPLANE(rc_zplane),
         .iPOLYMODE(zff polymode),
 2
                                                                                                                       2
         .iNULL PRIM(zff null prim),
                                                                                                                                .oMASK(rc mask),
         .iSTATE_ID(zff_state_var_indx),
                                                                                                                                .oBACK(rc back).
         . iST\_OFFSET\_FRONT\_EN (st\_poly\_offset\_front\_enable),
                                                                                                                                .oSTATE ID(rc state),
         . iST\_OFFSET\_BACK\_EN(st\_poly\_offset\_back\_enable),
                                                                                                                                .oCOVERED(rc covered),
         . iST\_OFFSET\_PARA\_EN (st\_poly\_offset\_para\_enable),
                                                                                                                                .oVALID(rc_send)
         . iST\_OFFSET\_FRONT (st\_poly\_offset\_front\_offset),
         . iST\_OFFSET\_BACK (st\_poly\_offset\_back\_offset),
         . iST\_SCALE\_FRONT(st\_poly\_offset\_front\_scale),
                                                                                                                              // !!!! rwr - tmp drivers until test can be updated
10
         . iST\_SCALE\_BACK (st\_poly\_offset\_back\_scale),
                                                                                                                              wire [13:0] tmp_st_cliprect_0_x_min = 'b0;
11
                                                                                                                      11
                                                                                                                              wire [13:0] tmp_st_cliprect_0_y_min = 'b0;
12
                                                                                                                              wire [13:0] tmp_st_cliprect_0_x_max = 'b0;
13
         .iEVENT(qm_event),
                                                                                                                              wire [13:0] tmp_st_cliprect_0_y_max = 'b0;
14
         .iEVENT_ID(qm_event_id),
                                                                                                                              wire [13:0] tmp_st_cliprect_1_x_min = 'b0;
15
         .iTILEX(qm tilex),
                                                                                                                      15
                                                                                                                              wire [13:0] tmp st cliprect 1 y min = 'b0;
16
         .iTILEY(qm tiley),
                                                                                                                      16
                                                                                                                               wire [13:0] tmp st cliprect 1 x max = 'b0;
17
         .iMASK(am quadmask).
                                                                                                                      17
                                                                                                                               wire [13:0] tmp_st_cliprect_1_y_max = 'b0;
18
         .iOM VALID(am quadmask valid).
                                                                                                                      18
                                                                                                                               wire [13:0] tmp_st_cliprect_2_x_min = 'b0;
         .iZ_MASK_NEEDED(qm_z_mask_needed),
19
                                                                                                                      19
                                                                                                                               wire [13:0] tmp st cliprect 2 y min = 'b0;
20
                                                                                                                      20
                                                                                                                               wire [13:0] tmp st cliprect 2 \times max = b0;
21
         .oEVENT(rc event).
                                                                                                                      21
                                                                                                                               wire [13:0] tmp st cliprect 2 y max = 'b0;
22
         .oTILEX(rc tilex).
                                                                                                                      22
                                                                                                                               wire [13:0] tmp_st_cliprect_3_x_min = 'b0;
23
         .oTILEY(rc tiley),
                                                                                                                      23
                                                                                                                               wire [13:0] tmp_st_cliprect_3_y_min = 'b0;
24
         .oMINZ(rc minz),
                                                                                                                      24
                                                                                                                               wire [13:0] tmp_st_cliprect_3_x_max = 'b0;
25
         .oMAXZ(rc_maxz),
                                                                                                                      25
                                                                                                                               wire [13:0] tmp_st_cliprect_3_y_max = 'b0;
                                           Page 65 of 87
                                                                                                                                                                  Page 66 of 87
                                                                               Ex. 2118 - sc.v
                                                                                                                                                                                                      Ex. 2118 - sc.v
                                                                                                                                .iRC_SC_HIER_MASK(rc_in_hier_mask),
2
       wire [2:0] tmp st msaa urc samp offset x = 'b0;
                                                                                                                                .iRC SC RB ID(rc in rb id),
                                                                                                                                .iRC_SC_SPLIT(rc_in_split),
       wire [2:0] tmp_st_msaa_urc_samp_offset_y = 'b0;
                                                                                                                                .iRC_SC_HIER_SEND(rc_in_hier_send),
4
       wire [2:0] tmp_st_msaa_llc_samp_offset_x = 'b0;
       wire [2:0] tmp_st_msaa_llc_samp_offset_y = 'b0;
                                                                                                                                .oSC RC HIER RTR(rc hier rtr),
       wire [2:0] tmp_st_msaa_lrc_samp_offset_x = 'b0;
       wire [2:0] tmp_st_msaa_lrc_samp_offset_y = 'b0;
                                                                                                                                .iFREEZE(qdpkr in fz),
       wire [127:0] tmp_st_aa_offset_tbl = 'b0;
                                                                                                                                .oQPP_SV_INDX(qpp_state_var_indx),
10
                                                                                                                      10
                                                                                                                                .iSV_MSAA_ENABLE(st_msaa_enable),
11
                                                                                                                      11
                                                                                                                                . iSV\_JSS\_ENABLE (st\_jss\_enable),
12
       sc_qdpr_proc usc_qdpr_proc(
                                                                                                                      12
                                                                                                                                .iSV_MSAA_NUM_SAMPLES(st_msaa_num_samples[2:0]),
13
         .iSCLK(sclk_sc),
                                                                                                                      13
                                                                                                                                .iSV_AA_MASK(st_aa_mask),
14
                                                                                                                      14
                                                                                                                                .iSV_JSS_X_DIM(st_jss_x_dim),
         .iSRST(sc_srst),
15
                                                                                                                                .iSV_JSS_Y_DIM(st_jss_y_dim),
16
         . iPRIM\_FIFO\_EMPTY (prim\_ff\_empty),
                                                                                                                                . iSV\_JSS\_SAMPLE\_SEL(\{st\_jss\_sample0\_sel[2:0],
17
         . iINTERP\_PRIM\_DATA (prim\_ff\_rd\_data),
                                                                                                                                          st_jss_sample1_sel[2:0],
18
         .oPRIM_FIFO_ADVANCE(prim_ff_re),
                                                                                                                                           st_jss_sample2_sel[2:0],
19
         .oCNTX0_DECR(cntx0_decr),
                                                                                                                                          st_jss_sample3_sel[2:0],
20
         .oCNTX1TO7 DECR(cntx1to7 decr),
                                                                                                                      20
                                                                                                                                          st jss sample4 sel[2:0].
21
                                                                                                                      21
                                                                                                                                          st jss sample5 sel[2:0].
22
         . iQM\_QUADMASK\_VALID (qm\_quadmask\_valid),
                                                                                                                      22
                                                                                                                                          st jss sample6 sel[2:0].
23
                                                                                                                      23
                                                                                                                                          st_jss_sample7_sel[2:0].
         .iOM DATA(tile ff wr data).
24
         .oTILEDATA FIFO FULL(tile ff full),
                                                                                                                      24
                                                                                                                                           st jss sample8 sel[2:0].
25
                                                                                                                                           st jss sample9 sel[2:0],
                                           Page 67 of 87
                                                                                                                                                                  Page 68 of 87
                                                                               Ex. 2118 - sc.v
                                                                                                                                                                                                      Ex. 2118 - sc.v
```

```
st\_jss\_sample10\_sel[2:0],
                                                                                                                                                                                  . oQPP\_PRIM\_DATA(qpp\_prim\_data),
                             st_jss_sample11_sel[2:0],
                                                                                                                                                                                  . oQPP\_FPOS\_EARLY (qpp\_fpos\_early),
                             st_jss_sample12_sel[2:0],
                                                                                                                                                                                   .oQPP_LAST_QDPAIR_OF_PRIM(qpp_last_qdpair_of_prim),
                             st_jss_sample13_sel[2:0],
                            st_jss_sample14_sel[2:0],
                                                                                                                                                                                  . oQPP\_Q0\_LAST\_OF\_TILE (qpp\_q0\_last\_of\_tile),
                             st\_jss\_sample15\_sel[2:0]\}),
             .iSV_CLIPRECT_ENABLE(st_cliprect_enable),
                                                                                                                                                                                  .oQPP\_Q0\_ZMASK\_NEEDED(qpp\_q0\_zmask\_needed),
             .iSV_CLIPRECT_RULE(st_clip_rule),
                                                                                                                                                                                  .oQPP_Q0_QHIT(qpp_q0_qhit),
             .iSV_CLIPRECT_0_XMIN(tmp_st_cliprect_0_x_min),
                                                                                                                                                                                  .oQPP_Q0_TILEX(qpp_q0_tilex),
             .iSV_CLIPRECT_0_YMIN(tmp_st_cliprect_0_y_min),
                                                                                                                                                                                  .oQPP_Q0_TILEY(qpp_q0_tiley),
11
                                                                                                                                                                      11
             .iSV_CLIPRECT_0_XMAX(tmp_st_cliprect_0_x_max),
                                                                                                                                                                                  .oQPP_Q0_QUADX(qpp_q0_quadx)
12
                                                                                                                                                                      12
             .iSV_CLIPRECT_0_YMAX(tmp_st_cliprect_0_y_max),
                                                                                                                                                                                  .oQPP_Q0_QUADY(qpp_q0_quady)
13
             .iSV_CLIPRECT_1_XMIN(tmp_st_cliprect_1_x_min),
                                                                                                                                                                      13
                                                                                                                                                                                  .oQPP_Q0_RB_ID(qpp_q0_rb_id)
14
             .iSV_CLIPRECT_1_YMIN(tmp_st_cliprect_1_y_min),
                                                                                                                                                                                  .oQPP Q0 SPLIT(qpp q0 split),
15
             .iSV_CLIPRECT_1_XMAX(tmp_st_cliprect_1_x_max),
                                                                                                                                                                      15
                                                                                                                                                                                  .oQPP_Q0_ULC_SAMPLE_MASK(qpp_q0_ulc_sample_mask),
16
             .iSV_CLIPRECT_1_YMAX(tmp_st_cliprect_1_y_max),
                                                                                                                                                                      16
                                                                                                                                                                                  . oQPP\_Q0\_URC\_SAMPLE\_MASK (qpp\_q0\_urc\_sample\_mask),
17
                                                                                                                                                                      17
             .iSV_CLIPRECT_2_XMIN(tmp_st_cliprect_2_x_min),
                                                                                                                                                                                  . oQPP\_Q0\_LLC\_SAMPLE\_MASK(qpp\_q0\_llc\_sample\_mask),
18
             .iSV_CLIPRECT_2_YMIN(tmp_st_cliprect_2_y_min),
                                                                                                                                                                      18
                                                                                                                                                                                  . oQPP\_Q0\_LRC\_SAMPLE\_MASK(qpp\_q0\_lrc\_sample\_mask), \\
19
             .iSV_CLIPRECT_2_XMAX(tmp_st_cliprect_2_x_max),
                                                                                                                                                                      19
                                                                                                                                                                                  . o QPP\_Q0\_ULC\_CNTRMOST\_SAMPLE\_ID (qpp\_q0\_ulc\_cntrmost\_sample\_id), and the property of the p
20
             .iSV_CLIPRECT_2_YMAX(tmp_st_cliprect_2_y_max),
                                                                                                                                                                      20
                                                                                                                                                                                  . o QPP\_Q0\_URC\_CNTRMOST\_SAMPLE\_ID (qpp\_q0\_urc\_cntrmost\_sample\_id), \\
21
             .iSV_CLIPRECT_3_XMIN(tmp_st_cliprect_3_x_min),
                                                                                                                                                                     21
                                                                                                                                                                                   .oQPP_Q0_LLC_CNTRMOST_SAMPLE_ID(qpp_q0_llc_cntrmost_sample_id),
22
             .iSV_CLIPRECT_3_YMIN(tmp_st_cliprect_3_y_min),
                                                                                                                                                                     22
                                                                                                                                                                                   .oQPP_Q0_LRC_CNTRMOST_SAMPLE_ID(qpp_q0_lrc_cntrmost_sample_id),
23
             .iSV_CLIPRECT_3_XMAX(tmp_st_cliprect_3_x_max),
24
             .iSV_CLIPRECT_3_YMAX(tmp_st_cliprect_3_y_max),
                                                                                                                                                                     24
                                                                                                                                                                                  .oQPP\_Q1\_LAST\_OF\_TILE(qpp\_q1\_last\_of\_tile),
25
                                                                                                                                                                                  . oQPP\_Q1\_ZMASK\_NEEDED(qpp\_q1\_zmask\_needed),
                                                             Page 69 of 87
                                                                                                              Ex. 2118 - sc.v
                                                                                                                                                                                                                                                                                    Ex. 2118 - sc.v
             . oQPP\_Q1\_QHIT(qpp\_q1\_qhit),
                                                                                                                                                                               assign qd0_data[`SC_QD_SPLIT]
                                                                                                                                                                                                                                             = qpp_q0_split;
 2
                                                                                                                                                                               assign\ qd0\_data[`SC\_QD\_ULC\_SAMPLE\_MASK]
             . oQPP\_Q1\_TILEX(qpp\_q1\_tilex),
                                                                                                                                                                                                                                                            = qpp_q0_ulc_sample_mask;
             .oQPP_Q1_TILEY(qpp_q1_tiley),
                                                                                                                                                                                assign qd0_data[`SC_QD_URC_SAMPLE_MASK]
                                                                                                                                                                                                                                                           = qpp_q0_urc_sample_mask;
             . oQPP\_Q1\_QUADX(qpp\_q1\_quadx),
                                                                                                                                                                                assign qd0_data[`SC_QD_LLC_SAMPLE_MASK]
                                                                                                                                                                                                                                                           = qpp_q0_llc_sample_mask;
             .oQPP\_Q1\_QUADY(qpp\_q1\_quady),
                                                                                                                                                                                assign\ qd0\_data[`SC\_QD\_LRC\_SAMPLE\_MASK]
                                                                                                                                                                                                                                                           = qpp_q0_lrc_sample_mask;
                                                                                                                                                                             \begin{array}{lll} assign & qd0\_data[\ SC\_QD\_ULC\_CNTRMOST\_SAMPLE\_ID] \\ qpp\_q0\_ulc\_cntrmost\_sample\_id; \end{array}
             . oQPP\_Q1\_RB\_ID(qpp\_q1\_rb\_id),
             .oQPP_Q1_SPLIT(qpp_q1_split),
                                                                                                                                                                              assign qd0_data[`SC_QD_URC_CNTRMOST_SAMPLE_ID] qpp_q0_urc_entrmost_sample_id;
             .oQPP_Q1_ULC_SAMPLE_MASK(qpp_q1_ulc_sample_mask),
             . o Q P P\_Q 1\_URC\_S A M P L E\_MASK (q p p\_q 1\_urc\_s ample\_mask),
                                                                                                                                                                             \begin{array}{ll} assign & qd0\_data[\ SC\_QD\_LLC\_CNTRMOST\_SAMPLE\_ID] \\ qpp\_q0\_llc\_cntrmost\_sample\_id; \end{array}
10
             . oQPP\_Ql\_LLC\_SAMPLE\_MASK(qpp\_ql\_llc\_sample\_mask),
                                                                                                                                                                                          qd0_data[`SC_QD_LRC_CNTRMOST_SAMPLE_ID]
11
             . oQPP\_Ql\_LRC\_SAMPLE\_MASK(qpp\_ql\_lrc\_sample\_mask),
                                                                                                                                                                      13
12
             .oQPP_Q1_ULC_CNTRMOST_SAMPLE_ID(qpp_q1_ulc_cntrmost_sample_id),
                                                                                                                                                                      14
             .oQPP_Q1_URC_CNTRMOST_SAMPLE_ID(qpp_q1_urc_cntrmost_sample_id),
                                                                                                                                                                      15
                                                                                                                                                                                assign qd1_data[`SC_QD_TILEX]
                                                                                                                                                                                                                                              = qpp\_q1\_tilex;
14
             .oQPP_Q1_LLC_CNTRMOST_SAMPLE_ID(qpp_q1_llc_cntrmost_sample_id),
                                                                                                                                                                      16
                                                                                                                                                                                assign qd1_data[`SC_QD_TILEY]
15
             . oQPP\_Q1\_LRC\_CNTRMOST\_SAMPLE\_ID(qpp\_q1\_lrc\_cntrmost\_sample\_id)
                                                                                                                                                                      17
                                                                                                                                                                                assign\ qd1\_data[`SC\_QD\_QUADX]
                                                                                                                                                                                                                                                = qpp_q1_quadx
16
                                                                                                                                                                      18
                                                                                                                                                                                assign\ qd1\_data[`SC\_QD\_QUADY]
17
                                                                                                                                                                      19
                                                                                                                                                                                 assign qd1_data[`SC_QD_RB_ID]
                                                                                                                                                                                                                                              = qpp\_q1\_rb\_id;
18
          wire ['SC QD DATA WIDTH-1:0] qd0 data;
                                                                                                                                                                                assign qd1_data[`SC_QD_SPLIT]
19
          wire ['SC_QD_DATA_WIDTH-1:0] qd1_data;
                                                                                                                                                                                assign qd1_data[`SC_QD_ULC_SAMPLE_MASK]
                                                                                                                                                                                                                                                            = qpp_ql_ulc_sample_mask;
20
                                                                                                                                                                     22
                                                                                                                                                                                assign qd1_data[`SC_QD_URC_SAMPLE_MASK]
                                                                                                                                                                                                                                                            = qpp_q1_urc_sample_mask;
21
          assign qd0_data[`SC_QD_TILEX]
                                                                        = qpp_q0_tilex;
                                                                                                                                                                     23
                                                                                                                                                                                assign qd1_data[`SC_QD_LLC_SAMPLE_MASK]
                                                                                                                                                                                                                                                           = qpp_q1_llc_sample_mask;
22
          assign qd0_data[`SC_QD_TILEY]
                                                                        = qpp\_q0\_tiley;
                                                                                                                                                                                assign qd1_data[`SC_QD_LRC_SAMPLE_MASK]
                                                                                                                                                                                                                                                         = qpp_q1_lrc_sample_mask;
23
          assign\ qd0\_data[`SC\_QD\_QUADX]
                                                                                                                                                                            \begin{array}{ll} assign & qd1\_data[\ SC\_QD\_ULC\_CNTRMOST\_SAMPLE\_ID] \\ qpp\_q1\_ulc\_cntrmost\_sample\_id; \end{array}
24
          assign qd0_data[`SC_QD_QUADY]
                                                                          = qpp_q0_quady;
                                                                                                                                                                                assign qd1 data['SC QD URC CNTRMOST SAMPLE ID]
          assign qd0_data[`SC_QD_RB_ID]
                                                                        = qpp\_q0\_rb\_id;
                                                                                                                                                                             qpp ql urc_cntrmost_sample_id;
                                                            Page 71 of 87
                                                                                                                                                                                                                                  Page 72 of 87
                                                                                                              Ex. 2118 - sc.v
                                                                                                                                                                                                                                                                                    Ex. 2118 - sc.v
```

```
\begin{array}{lll} assign & qd1\_data[`SC\_QD\_LLC\_CNTRMOST\_SAMPLE\_ID] \\ qpp\_q1\_llc\_cntrmost\_sample\_id; \end{array}
                                                                                                                                        .early\_fpos(qpp\_fpos\_early),
                                                                                                                               2
                                                                                                                                        .primdata(qpp_prim_data),
      assign = qd1\_data["SC\_QD\_LRC\_CNTRMOST\_SAMPLE\_ID] \\ qpp\_q1\_lrc\_entrmost\_sample\_id;
                                                                                                                                        . iterator\_input\_fz (iterator\_input\_fz), \\
                                                                                                                                        .detail_mask_accum_rdy(detail_mask_accum_rdy),
       sc_packer usc_packer(
         .sclk(sclk_sc),
                                                                                                                                        .qdpkr_in_fz(qdpkr_in_fz),
          .srst(sc srst),
10
                                                                                                                                        .pkr_qd0(pkr_qd0),
11
          .sp0_disable(ROM_SP0_disable),
                                                                                                                                        .pkr_qd1(pkr_qd1),
                                                                                                                                        .pkr_qd2(pkr_qd2),
12
          .sp1 disable(ROM SP1 disable),
13
          .sp2 disable(ROM SP2 disable),
                                                                                                                                        .pkr_qd3(pkr_qd3),
                                                                                                                              13
                                                                                                                                        .pkr_qdhit0(pkr_qdhit0),
14
          .sp3 disable(ROM SP3 disable),
                                                                                                                                        .pkr_qdhit1(pkr_qdhit1),
15
                                                                                                                              15
                                                                                                                                        .pkr qdhit2(pkr qdhit2).
16
          .qd0\_hit(qpp\_q0\_qhit),
                                                                                                                              16
                                                                                                                                        .pkr qdhit3(pkr qdhit3),
17
          .qd0\_lqt(qpp\_q0\_last\_of\_tile),
                                                                                                                              17
                                                                                                                                        .pkr primdata(pkr_primdata),
18
          .qd0\_zneeded(qpp\_q0\_zmask\_needed),
                                                                                                                              18
                                                                                                                                        .pkr ds one clk command(pkr ds one clk command),
19
          .qd0\_data(qd0\_data),
20
                                                                                                                              19
                                                                                                                                        .pkr ds end of prim(pkr ds end of prim).
21
          .qd1_hit(qpp_q1_qhit),
                                                                                                                              20
                                                                                                                                        .pkr ds end of vector(pkr ds end of vector),
                                                                                                                              21
22
                                                                                                                                        .pkr_send_row(pkr_send_row),
          .qd1_lqt(qpp_q1_last_of_tile),
23
          .qd1_zneeded(qpp_q1_zmask_needed),
                                                                                                                              22
24
          .qd1_data(qd1_data),
                                                                                                                              23
                                                                                                                                        .pkr_sv_indx(pkr_sv_indx),
25
                                                                                                                              24
                                                                                                                                        .pkr_cntx0_busy(pkr_cntx0_busy),
                                                                                                                              25
                                                                                                                                        .pkr_cntx1to7_busy(pkr_cntx1to7_busy),
          .last_qdpair_of_prim(qpp_last_qdpair_of_prim),
                                              Page 73 of 87
                                                                                                                                                                            Page 74 of 87
                                                                                    Ex. 2118 - sc.v
                                                                                                                                                                                                                  Ex. 2118 - sc.v
                                                                                                                                        .oDETAIL_MASK(detail_mask),
 2
         //control signals to the detail mask accumulater
                                                                                                                               2
                                                                                                                                        .oDETAIL MASK VALID(detail mask valid).
          .detail hit 0(detail hit 0),
                                                                                                                                        .iRDY_FOR_DETAIL_MASK(rdy_for_detail_mask)
          .detail lqt 0(detail lqt 0),
          .detail hit 1(detail hit 1).
                                                                                                                               5
         .detail\_lqt\_l(detail\_lqt\_l)
 7
                                                                                                                                      sc_iter usc_iter (
                                                                                                                                        .sclk(sclk sc),
                                                                                                                                        .srst(sc_srst),
        sc_detail_mask_accum usc_detail_mask_accum (
10
                                                                                                                              10
                                                                                                                                        .st_send_centers(st_send_centers),
11
          .iSCLK(sclk_sc),
                                                                                                                              11
                                                                                                                                        .st\_send\_centroids(st\_send\_centroids),
12
          .iSRST(sc_srst),
                                                                                                                              12
                                                                                                                                        .st\_output\_screen\_xy(st\_output\_screen\_xy),\\
13
                                                                                                                                        .st_iter_msaa_enable(st_iter_msaa_enable),
14
          .oRDY_FOR_TILE_DATA(detail_mask_accum_rdy),
                                                                                                                              14
                                                                                                                                        .st_iter_msaa_num_samples(st_iter_msaa_num_samples),
15
                                                                                                                              15
                                                                                                                                        .st_iter_jss_enable(st_iter_jss_enable),
16
          .iX_0(qd0_data[`SC_QD_QUADX]),
17
          . iY\_0 (qd0\_data[`SC\_QD\_QUADY]),
                                                                                                                                        .pkr_qd0(pkr_qd0),
18
          .iHIT_0(detail_hit_0),
                                                                                                                              18
                                                                                                                                        .pkr_qd1(pkr_qd1),
19
          .iEOT_0(detail_lqt_0),
                                                                                                                                        .pkr_qd2(pkr_qd2),
20
                                                                                                                              20
                                                                                                                                        .pkr qd3(pkr qd3),
21
          .iX_1(qd1_data[`SC_QD_QUADX]),
                                                                                                                              21
                                                                                                                                        .pkr qdhit0(pkr qdhit0).
22
          .iY_1(qd1_data[`SC_QD_QUADY]),
                                                                                                                              22
                                                                                                                                        .pkr qdhit1(pkr qdhit1),
23
          .iHIT_1(detail_hit_1),
                                                                                                                              23
                                                                                                                                        .pkr qdhit2(pkr qdhit2),
24
          .iEOT 1(detail lqt 1),
                                                                                                                              24
                                                                                                                                        .pkr_qdhit3(pkr_qdhit3),
25
                                                                                                                                        .pkr_primdata(pkr_primdata).
                                              Page 75 of 87
                                                                                                                                                                            Page 76 of 87
                                                                                    Ex. 2118 - sc.v
                                                                                                                                                                                                                  Ex. 2118 - sc.v
```

```
.pkr ds one clk command(pkr ds one clk command),
                                                                                                                                 .iterator SX1 quad tilex(iterator SX1 quad tilex),
 2
         .pkr_ds_end_of_prim(pkr_ds_end_of_prim),
                                                                                                                        2
                                                                                                                                 iterator SX1 quad tiley(iterator SX1 quad tiley),
         .pkr_ds_end_of_vector(pkr_ds_end_of_vector),
                                                                                                                                 iterator SX1 quad rb id(iterator SX1 quad rb id),
         .pkr_send_row(pkr_send_row),
                                                                                                                                 . iterator\_SX1\_quad\_split (iterator\_SX1\_quad\_split),
                                                                                                                                 .iterator_SX1_quad_send(iterator_SX1_quad_send),
         .pkr_cntx0_busy(pkr_cntx0_busy),
                                                                                                                                 .SX1_iterator_quad_rtr(SX1_iterator_quad_rtr),
         .pkr\_cntx1to7\_busy(pkr\_cntx1to7\_busy),
         .pkr\_iter\_cntx0\_busy(pkr\_iter\_cntx0\_busy),
                                                                                                                                 .iterator_SQ_data(iterator_SQ_data),
         .pkr_iter_cntx1to7_busy(pkr_iter_cntx1to7_busy),
                                                                                                                                 .iterator_SQ_valid(iterator_SQ_valid),
10
                                                                                                                                 . SQ\_iterator\_free\_buff (SQ\_iterator\_free\_buff),
11
         .iterator_sv_indx(iterator_sv_indx),
                                                                                                                                 .SQ_iterator_dec_cntr_cnt(SQ_iterator_dec_cntr_cnt),
12
         . iterator\_input\_fz (iterator\_input\_fz), \\
13
                                                                                                                       13
                                                                                                                                 .iterator_SP0_data(iterator_SP0_data),
         .iterator SX0 quad x(iterator SX0 quad x),
14
                                                                                                                                 .iterator_SP0_type(iterator_SP0_type),
         .iterator SX0 quad y(iterator SX0 quad y),
15
                                                                                                                       15
                                                                                                                                 .iterator SP0 last quad(iterator SP0 last quad),
         .iterator SX0 quad mask(iterator SX0 quad mask),
16
         .iterator SX0 quad tilex(iterator SX0 quad tilex),
                                                                                                                       16
                                                                                                                                 .iterator SP0 valid(iterator SP0 valid),
17
                                                                                                                       17
         .iterator SX0 quad tilev(iterator SX0 quad tilev).
18
         iterator SX0 quad rb id(iterator SX0 quad rb id).
                                                                                                                       18
                                                                                                                                 .iterator SP1 data(iterator SP1 data).
19
         .iterator SX0 quad split(iterator SX0 quad split),
                                                                                                                       19
                                                                                                                                 .iterator SP1 type(iterator SP1 type),
                                                                                                                                 .iterator SP1_last_quad(iterator_SP1_last_quad),
20
         .iterator SX0 quad send(iterator SX0 quad send),
                                                                                                                       20
21
         .SX0 iterator quad rtr(SX0 iterator quad rtr),
                                                                                                                       21
                                                                                                                                 .iterator SP1 valid(iterator SP1 valid),
22
                                                                                                                       22
23
         .iterator SX1 quad x(iterator SX1 quad x),
                                                                                                                       23
                                                                                                                                 .iterator SP2 data(iterator SP2 data),
24
         .iterator_SX1_quad_y(iterator_SX1_quad_y),
                                                                                                                       24
                                                                                                                                 .iterator_SP2_type(iterator_SP2_type),
25
         .iterator_SX1_quad_mask(iterator_SX1_quad_mask),
                                                                                                                       25
                                                                                                                                 .iterator_SP2_last_quad(iterator_SP2_last_quad),
                                            Page 77 of 87
                                                                                                                                                                   Page 78 of 87
                                                                               Ex. 2118 - sc.v
                                                                                                                                                                                                       Ex. 2118 - sc.v
         .iterator_SP2_valid(iterator_SP2_valid),
                                                                                                                                             PA_SC_p4}; // [31:0]
 2
         iterator SP3 data(iterator SP3 data).
                                                                                                                        3 // Assign registered inputs for use in sc_stage_reg
 4
         .iterator SP3 type(iterator SP3 type),
                                                                                                                               assign valid_in = pa_sc_inputs_reg[270];
         .iterator_SP3_last_quad(iterator_SP3_last_quad),
                                                                                                                               assign\ v0\_indx\_in = pa\_sc\_inputs\_reg[269:268];
         .iterator SP3 valid(iterator SP3 valid)
                                                                                                                               assign\ cntl\_in = pa\_sc\_inputs\_reg[267:238];
7
                                                                                                                                assign phase_in = pa_sc_inputs_reg[237:236];
                                                                                                                               assign zminmax_in = pa_sc_inputs_reg[235:222];
                                                                                                                                assign xy0_in = pa_sc_inputs_reg[221:204];
10 // Assign statements
                                                                                                                       10
                                                                                                                               assign xy1_in = pa_sc_inputs_reg[203:186];
11
                                                                                                                       11
                                                                                                                               assign xy2_in = pa_sc_inputs_reg[185:168];
12
                                                                                                                       12
                                                                                                                              assign p0_in = pa_sc_inputs_reg[167:136];
13 // Concatenate inputs to register them
                                                                                                                              assign p1_in = pa_sc_inputs_reg[135:96];
14
       assign pa_sc_inputs = {PA_SC_valid, // [270]
                                                                                                                              assign p2_in = pa_sc_inputs_reg[95:64];
15
                   PA_SC_v0_indx, // [269:268]
                                                                                                                              assign p3_in = pa_sc_inputs_reg[63:32];
                    PA_SC_cntl, // [267:238]
                                                                                                                               assign p4_in = pa_sc_inputs_reg[31:0];
17
                   PA_SC_phase, // [237:236]
18
                   PA_SC_zminmax, // [235:222]
                                                                                                                       18 // Concatenate some of outputs of sc_stage_reg to create write data for
19
                   PA_SC_xy0, // [221:204]
                                                                                                                       19 // primitive fifo.
                   PA_SC_xy1, // [203:186]
20
                                                                                                                       20
                                                                                                                              assign prim ff wr data['SC PI EVENT]
                                                                                                                                                                             = sr event;
                   PA_SC_xy2, //[185:168]
21
                                                                                                                              assign prim ff wr data[`SC PI EVENTID] = sr event id;
                                                                                                                       21
                   PA_SC_p0, // [167:136]
22
                                                                                                                              assign prim_ff_wr_data[`SC_PI_DEALLOCATE_SLOT] = sr_dealloc_slot;
                                                                                                                       22
23
                    PA_SC_p1, // [135:96]
                                                                                                                       23
                                                                                                                               assign prim_ff_wr_data[`SC_PI_FIRST_PRIM_OF_SLOT] = sr_first_prim_of_slot;
24
                    PA_SC_p2, // [95:64]
                                                                                                                       24
                                                                                                                               assign prim ff wr data['SC PI END OF PACKET] = sr end of pkt;
                    PA_SC_p3, // [63:32]
                                                                                                                                assign prim_ff_wr_data[`SC_PI_BACK_FACE] = sr_back_face;
                                            Page 79 of 87
                                                                                                                                                                   Page 80 of 87
                                                                               Ex. 2118 - sc.v
                                                                                                                                                                                                       Ex. 2118 - sc.v
```

```
assign prim ff wr data['SC PI PROVOKING VERT] = sr provoking vertex;
                                                                                                                                                     // [359:357] 3
                                                                                                               1 //
                                                                                                                                   sr prim type,
                                                                                                              2 //
2
       assign prim ff wr data['SC PI STATE] = sr state var indx;
                                                                                                                                   sr ref x,
                                                                                                                                                    // [356:339] 18
       assign prim ff wr data[`SC PI TYPE]
                                               = sr_prim_type;
                                                                                                              3 //
                                                                                                                                   sr ref y,
                                                                                                                                                    // [338:321] 18
       assign prim_ff_wr_data[`SC_PI_V0X] = sr_ref_x;
                                                                                                                                   sr_param_cache_indx0, // [320:310] 11
       assign \ prim\_ff\_wr\_data[`SC\_PI\_V0Y] \qquad \qquad = sr\_ref\_y;
                                                                                                               5 //
                                                                                                                                   sr_param_cache_indx1, // [309:297] 11
       assign prim_ff_wr_data[`SC_PI_PC_PTR0] = sr_param_cache_indx0;
                                                                                                               6 //
                                                                                                                                   sr_param_cache_indx2, // [298:288] 11
       assign\ prim\_ff\_wr\_data[`SC\_PI\_PC\_PTR1] \\ \hspace*{0.5in} = sr\_param\_cache\_indx1;
                                                                                                                                   sr_i0,
                                                                                                                                               // [287:256] 32
                                                                                                                                               // [255:224] 32
       assign\ prim\_ff\_wr\_data[`SC\_PI\_PC\_PTR2] \\ \hspace*{0.5in} = sr\_param\_cache\_indx2;
                                                                                                                                   sr_ix,
                                                                                                              9 //
                                                                                                                                               // [223:192] 32
       assign prim_ff_wr_data[`SC_PI_IW0]
                                                                                                                                   sr_iy,
                                                                                                                                               // [191:160] 32
       assign prim_ff_wr_data[`SC_PI_IW_DX]
                                                 = sr ix;
                                                                                                                                   sr_j0,
11
       assign prim_ff_wr_data[`SC_PI_IW_DY]
                                                 = sr_iy;
                                                                                                                                                  // [159:128] 32
                                                                                                                                   sr_jx,
12
       assign prim_ff_wr_data[`SC_PI_JW0]
                                               = sr_j0;
                                                                                                                                   sr_jy,
                                                                                                                                                 // [127:96] 32
                                                 = sr_jx;
13
                                                                                                                                                 // [95:64] 32
       assign prim ff wr data['SC PI JW DX]
                                                                                                                                   sr w0,
14
       assign prim_ff_wr_data[`SC_PI_JW_DY]
                                                 = sr_jy;
                                                                                                                                                 // [63:32] 32
                                                                                                                                   SF WX,
                                                                                                                                                   // [31:0] 32
15
                                                  = sr w0;
                                                                                                             15 //
       assign prim ff wr data['SC PI INVW0]
                                                                                                                                   sr wy);
                                                 = sr_wx;
                                                                                                                                                // total = 373 bits
16
       assign prim ff wr data['SC PI INVW DX]
                                                                                                             16
17
                                                                                                             17
       assign prim ff wr data['SC PI INVW DY]
                                                 = sr wv:
18
                                                                                                             18 // Concatenate some of outputs of sc stage reg to create write data for Z fifo
19 // assign prim ff wr data = {sr event id,
                                                // [372:369] 4
                                                                                                             19
                                                                                                                   assign z ff wr data['SC ZD NULL PRIM] = sr null prim zff;
20 //
                    sr dealloc slot // [368] 1
                                                                                                             20
                                                                                                                    assign z ff wr data[ SC ZD ZY MAX] = sr zv max zff
21 //
                     sr_first_prim_of_slot, // [367] 1
                                                                                                             21
                                                                                                                    assign z ff wr data['SC ZD BACK FACE] = sr back face zff:
22 //
                    sr end of pkt, // [366] 1
                                                                                                             22
                                                                                                                    assign z ff wr data[SC ZD PRIM TYPE] = sr prim type zff;
23 //
                     sr back face,
                                       // [365] 1
                                                                                                             23
                                                                                                                     assign z ff wr data['SC ZD POLYMODE] = sr polymode zff;
24 //
                     sr_provoking_vertex, // [364:363] 2
                                                                                                             24
                                                                                                                     assign z_ff_wr_data[`SC_ZD_STATE_INDX] = sr_state_var_indx_zff;
25 //
                     sr_state_var_indx, // [362:360] 3
                                                                                                                     assign z_ff_wr_data[`SC_ZD_REF_X] = sr_ref_x_zff;
                                        Page 81 of 87
                                                                                                                                                     Page 82 of 87
                                                                         Ex. 2118 - sc.v
                                                                                                                                                                                      Ex. 2118 - sc.v
       assign z_ff_wr_data[`SC_ZD_REF_Y] = sr_ref_y_zff;
                                                                                                                     assign tile_ff_wr_data[`SC_TD_YMAX] = qm_ymax;
2
       assign z ff wr data[`SC ZD Z MIN] = sr z min zff;
                                                                                                                     assign tile ff wr data['SC TD BBFRACTBITS] = qm bb fract bits;
       assign z ff wr data[SC ZD Z MAX] = sr z max zff;
                                                                                                                     assign tile ff wr data['SC TD XDIR] = qm xdir;
       assign z ff wr data[ SC ZD Z0] = sr z0 zff:
                                                                                                                     assign tile ff wr data['SC TD YDIR] = qm ydir;
       assign z ff wr data['SC ZD ZX] = sr zx zff;
                                                                                                                     assign tile_ff_wr_data['SC_TD_TILEX] = qm_tilex;
       assign z_ff_wr_data[`SC_ZD_ZY] = sr_zy_zff;
                                                                                                                     assign tile_ff_wr_data['SC_TD_TILEY] = qm_tiley;
                                                                                                                     assign\ tile\_ff\_wr\_data[`SC\_TD\_TILEX\_M3] = qm\_tilex\_m3;
 8 // assign z_ff_wr_data = {sr_polymode, // [171] 1
                                                                                                                     assign tile ff wr data['SC TD TILEY M3] = qm tiley m3;
 9 //
                 sr_state_var_indx, //[170:168] 3
                                                                                                                     assign tile_ff_wr_data[`SC_TD_XMAJOR] = qm_xmajor;
10 //
                   sr_ref_x, // [167:150] 18
                                                                                                                     assign\ tile\_ff\_wr\_data[`SC\_TD\_E0\_SAMPLE] = qm\_e0;
11 //
                   sr_ref_y,
                                  // [149:132] 18
                                                                                                                     assign tile_ff_wr_data[`SC_TD_El_SAMPLE] = qm_el;
12 //
                   sr z min,
                                // [131:118] 14
                                                                                                                     assign tile_ff_wr_data[`SC_TD_E2_SAMPLE] = qm_e2;
                                     // [117:104] 14
                  sr_z_max,
                                                                                                                     assign tile_ff_wr_data[`SC_TD_E0_DX] = qm_dxe0;
                                   // [103:72] 32
                   sr_z0,
                                                                                                                     assign tile_ff_wr_data[`SC_TD_E0_DY] = qm_dye0
                                   // [71:36] 36
                   Sr_zx,
                                                                                                                    assign tile_ff_wr_data[`SC_TD_E1_DX] = qm_dxe1;
                                    // [35:0] 36
                                                                                                                    assign tile_ff_wr_data[`SC_TD_E1_DY] = qm_dye1;
                    sr_zy};
17
                                // total = 172 bits
                                                                                                                    assign tile_ff_wr_data[`SC_TD_E2_DX] = qm_dxe2;
18
                                                                                                                    assign tile_ff_wr_data[`SC_TD_E2_DY] = qm_dye2;
19 // Concatenate outputs of sc_quadmask to create write data for tile fifo.
                                                                                                                     assign tile_ff_wr_data[`SC_TD_STIPPLE_MASK] = 8'b111111111;
      assign tile ff wr data['SC TD LAST TILE] = qm last tile;
                                                                                                             20
20
                                                                                                                                                            // [265] 1
21
      assign tile ff wr data['SC TD ZMASK NEEDED] = qm z mask needed;
                                                                                                             21 // assign tile ff wr data = {qm last tile,
22
                                                                                                             22 //
                                                                                                                                                   // [264] 1
       assign tile ff wr data['SC TD EVENT] = qm event:
                                                                                                                                   qm event,
23
                                                                                                             23 //
                                                                                                                                                    // [263] 1
       assign tile ff wr data['SC TD XMIN] = qm xmin;
                                                                                                                                   qm xdir,
24
                                             = qm xmax;
                                                                                                             24 //
       assign tile ff wr data['SC TD XMAX]
                                                                                                                                  qm ydir,
                                                                                                                                                    // [262] 1
       assign tile_ff_wr_data[`SC_TD_YMIN] = qm_ymin;
                                                                                                             25 //
                                                                                                                                   gm z mask needed, // [261] 1
                                                                                                                                                     Page 84 of 87
                                        Page 83 of 87
                                                                                                                                                                                      Ex. 2118 - sc.v
                                                                         Ex. 2118 - sc.v
```

```
assign\ zff\_ref\_y = z\_ff\_rd\_data[`SC\_ZD\_REF\_Y];
 1 //
                      gm bb fract bits, // [260:257] 4
 2 //
                      qm_tilex,
                                 // [256:247] 10
                                                                                                                   assign zff_z_min = z_ff_rd_data[`SC_ZD_Z_MIN];
                                                                                                                    assign\ zff\_z\_max = z\_ff\_rd\_data[`SC\_ZD\_Z\_MAX];
 3 //
                      qm_tiley,
                                     // [246:237] 10
 4 //
                      qm_xmin,
                                       // [236:234] 3
                                                                                                                    assign zff_z0 = z_ff_rd_data[`SC_ZD_Z0];
 5 //
                      qm_xmax,
                                       // [233:231] 3
                                                                                                                    assign\ zff\_zx = z\_ff\_rd\_data[`SC\_ZD\_ZX];
 6 //
                      qm_ymin,
                                       // [230:228] 3
                                                                                                                    assign\ zff\_zy = z\_ff\_rd\_data[`SC\_ZD\_ZY];
                                       // [227:225] 3
                     qm_ymax,
 8 //
                                      // [224:188] 37
                     qm_e0,
                                      // [187:151] 37
                      qm_e1,
                                      // [150:114] 37
                                                                                                             10 // Group aa sample offsets
                     qm_e2,
11 //
                     qm_dxe0,
                                       // [113:95] 19
                                                                                                             11 // assign st_aa_offset_tbl = {
12 //
                                       // [94:76] 19
                     qm_dye0,
                                                                                                                       st_sample_15_y,
13 //
                                       // [75:57] 19
                                                                                                                         st_sample_15_x,
                      qm dxe1,
                                       // [56:38] 19
                                                                                                                         st_sample_14_y,
                      qm dye1,
15 //
                                       // [37:19] 19
                                                                                                             15 //
                                                                                                                          st sample 14 x,
                      qm dxe2,
16 //
                                       // [18:0] 19
                                                                                                             16 //
                                                                                                                          st sample 13 v,
                      qm_dye2};
17
                                // total = 266 bits
                                                                                                             17 //
                                                                                                                          st sample 13 x.
                                                                                                             18 //
                                                                                                                          st_sample_12_y,
18 // Assign read data of z fifo
                                                                                                             19 //
19
      assign zff null prim = z ff rd data['SC ZD NULL PRIM];
                                                                                                                          st sample 12 x,
20
                                                                                                             20 //
                                                                                                                          st sample 11 v,
       assign zff zv max = z ff rd data[SC ZD ZY MAX];
21
                                                                                                             21 //
      assign zff_back_face = z_ff_rd_data[`SC_ZD_BACK_FACE];
                                                                                                                          st sample 11 x,
22
      assign zff_prim_type = z_ff_rd_data[`SC_ZD_PRIM_TYPE];
                                                                                                             22 //
                                                                                                                          st sample 10 y,
23
       assign zff_polymode = z_ff_rd_data[`SC_ZD_POLYMODE];
                                                                                                             23 //
                                                                                                                          st_sample_10_x,
24
       assign\ zff\_state\_var\_indx = z\_ff\_rd\_data[`SC\_ZD\_STATE\_INDX];
                                                                                                             24 //
                                                                                                                          st_sample_9_y,
25
       assign zff_ref_x = z_ff_rd_data[`SC_ZD_REF_X];
                                                                                                             25 //
                                                                                                                          st_sample_9_x,
                                        Page 85 of 87
                                                                                                                                                     Page 86 of 87
                                                                         Ex. 2118 - sc.v
                                                                                                                                                                                     Ex. 2118 - sc.v
             st_sample_8_y,
 2 //
             st sample 8 x,
             st sample 7 y,
             st sample 7 x,
             st_sample_6_y,
             st_sample_6_x,
             st_sample_5_y,
             st_sample_5_x,
             st_sample_4_y,
10 //
             st_sample_4_x,
11 //
             st_sample_3_y,
12 //
             st_sample_3_x,
13 //
             st_sample_2_y,
14 //
             st_sample_2_x,
15 //
             st_sample_l_y,
16 //
             st_sample_l_x,
17 //
             st_sample_0_y,
18 //
             st_sample_0_x
19 // };
20
21 endmodule // sc
22
                                        Page 87 of 87
                                                                        Ex. 2118 - sc.v
```