[redacted]

ORIGINATE DATE	EDIT DATE	DOCUMENT-REV. NUM.	PAGE
14 August, 200114 August, 20017 May	4 September, 201524 August, 20016 August	GEN-CXXXXX-REVA	1 of 20

Author: Laurent Lefebvre

Issue To: **Copy No:**

R400 Sequencer Specification

SEQ

Version 0.42

Overview: This is an architectural specification for the R400 Sequencer block (SEQ). It provides an overview of the required capabilities and expected uses of the block. It also describes the block interfaces, internal sub-blocks, and provides internal state diagrams.

AUTOMATICALLY UPDATED FIELDS:
Document Location: C:\perforce\r400\arch\doc\gfx\RE\R400_Sequencer.doc
Current Intranet Search Title: R400 Sequencer Specification

APPROVALS

Name/Dept	Signature/Date

Remarks:

THIS DOCUMENT CONTAINS [redacted] INFORMATION THAT COULD BE SUBSTANTIALLY DETRIMENTAL TO THE INTEREST OF ATI TECHNOLOGIES INC. THROUGH UNAUTHORIZED USE OR DISCLOSURE.

ATI 2010
LG v. ATI
IPR2015-00325

AMD1044_0016660

[redacted]

Table Of Contents

[redacted]

Revision Changes:

Rev 0.1 (Laurent Lefebvre) Date: May 7, 2001	First draft.
Rev 0.2 (Laurent Lefebvre) Date : July 9, 2001	Changed the interfaces to reflect the changes in the SP. Added some details in the arbitration section.
Rev 0.3 (Laurent Lefebvre) Date : August 6, 2001	Reviewed the Sequencer spec after the meeting on August 3, 2001.
Rev 0.4 (Laurent Lefebvre) Date : August 24, 2001	Added the dynamic allocation method for register file and an example (written in part by Vic) of the flow of pixels/vertices in the sequencer.

1. Overview

The sequencer first arbitrates between vectors of 16 ~~(maybe 32)~~ vertices that arrive directly from primitive assembly and vectors of ~~8~~4 quads (16 pixels) ~~(32 pixels)~~ that are generated in the raster engine.

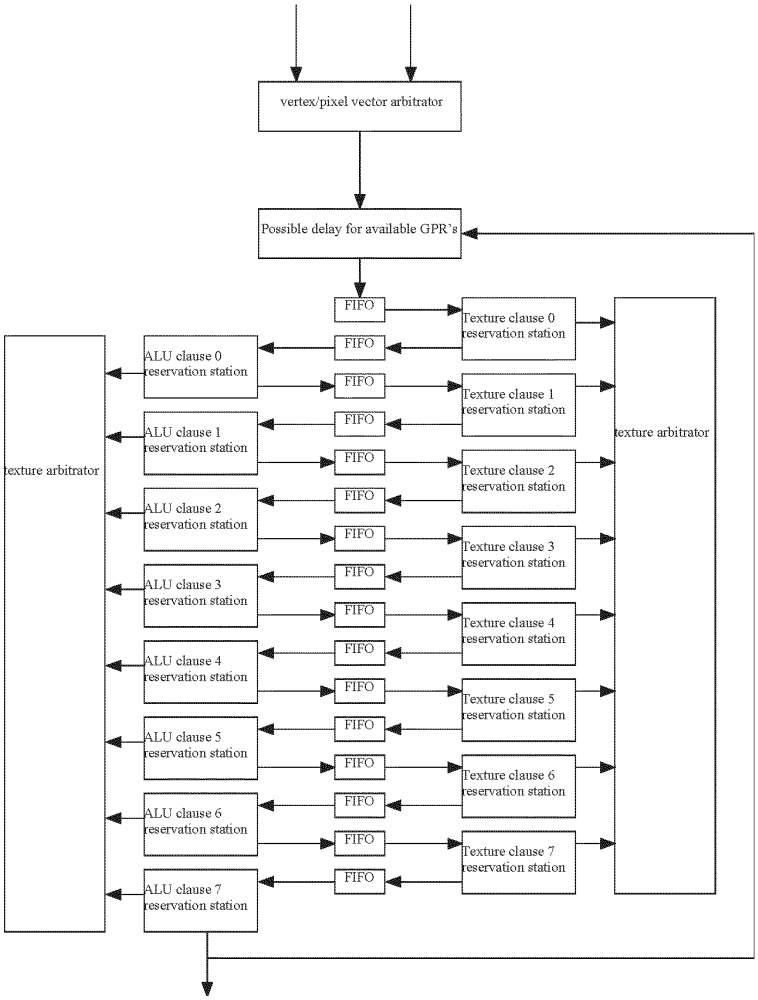
The vertex or pixel program specifies how many GPR's it needs to execute. The sequencer will not start the next vector until the needed space is available.

The sequencer is based on the R300 design. It chooses ~~an~~ two ALU clauses and a texture clause to execute, and executes all of the instructions in ~~a~~a clause before looking for a new clause of the same type. Two ALU clauses are executed interleaved to hide the ALU latency. Each vector will have eight texture and eight ALU clauses, but clauses do not need to contain instructions. A vector of pixels or vertices ping-pongs along the sequencer FIFO, bouncing from texture reservation station to alu reservation station. A FIFO exists between each reservation stage, holding up vectors until the vector currently occupying a reservation station has left. A vector at a reservation station can be chosen to execute. The sequencer looks at all eight alu reservation stations to choose an alu clause to execute and all eight texture stations to choose a texture clause to execute. The arbitrator will give priority to clauses/reservation stations closer to the ~~top~~ bottom of the pipeline. It will not execute an alu clause until the texture fetches initiated by the previous texture clause have completed. There are two separate sets of reservation stations, one for pixel vectors and one for vertices vectors. This way a pixel can pass a vertex and a vertex can pass a pixel.

To support the shader pipe the raster engine also contains the shader instruction cache and constant store. There are only one constant store for the whole chip and one instruction store. These will be shared among the four shader pipes.

[redacted]

1.1 Top Level Block Diagram



There are two sets of the above figure, one for vertices and one for pixels.

The rasterizer always checks the vertices FIFO first and if allowed by the sequencer sends the data to the shader. If the vertex FIFO is empty then, the rasterizer takes the first entry of the pixel FIFO (a vector of ~~32~~ 16 pixels) and sends it to the interpolators. Then the sequencer takes control of the packet. The packet consists of 3 bits of state, 6-7 bits for the base address of the Shader program and some information on the coverage to determine texture LOD. All other information (2x2 adresses) is put in a FIFO (one for the pixels and one for the vertices) and retrieved when the packet finishes its last clause.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.