February 4, 2020

Microsoft Corp. v. Directstream, LLC

IPR2018-01601, -01602, -01603

U.S. Patent No. 7,225,324

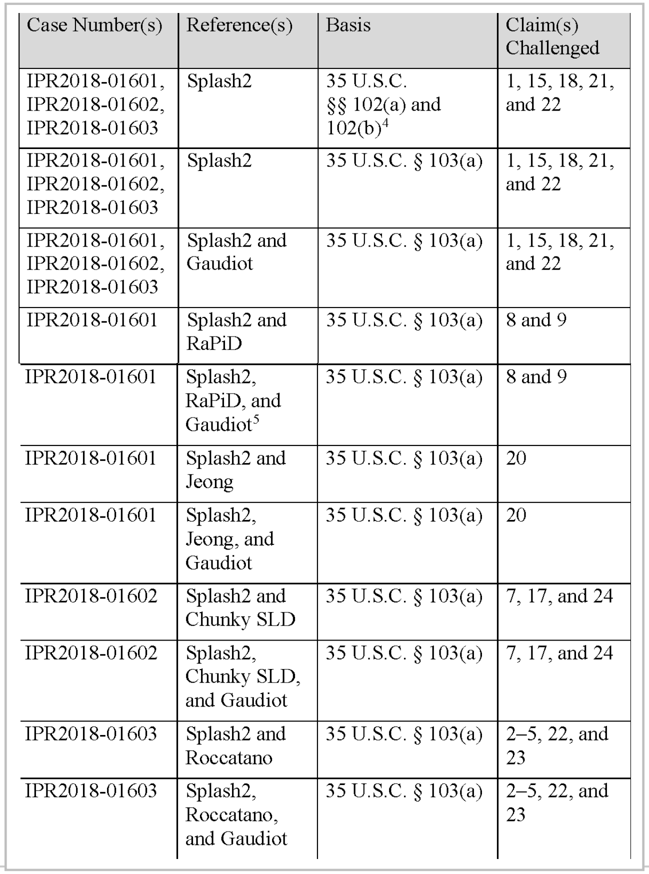
IPR2018-01605, -01606, -01607

U.S Patent No. 7,620,800

SIDLEY

Grounds

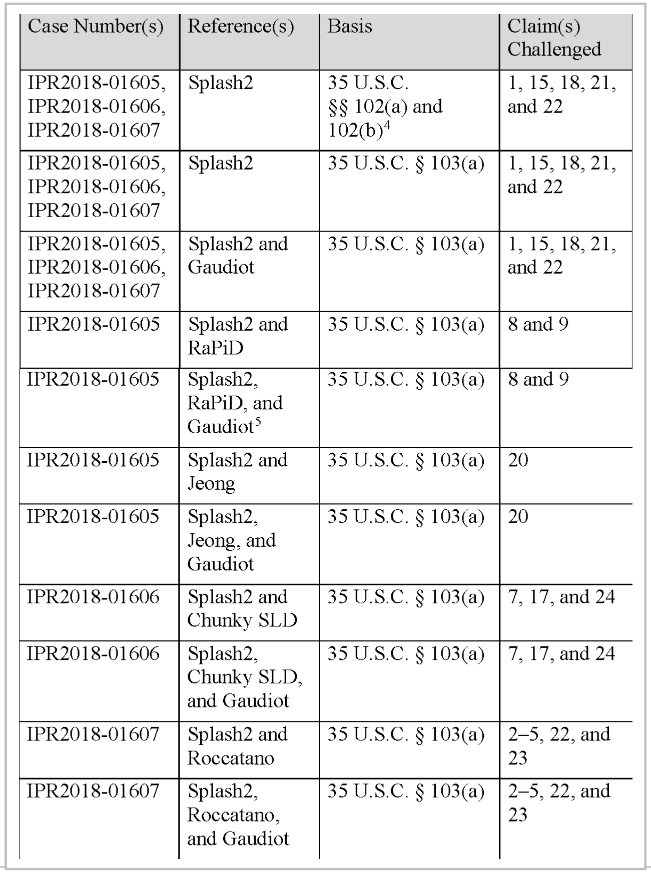
IPR2018-01601: Trial of 324 Patent



Case Number(s)	Reference(s)	Basis	Claim(s) Challenged
IPR2018-01601, IPR2018-01602, IPR2018-01603	Splash2	35 U.S.C. §§ 102(a) and 102(b)[4]	1, 15, 18, 21, and 22
IPR2018-01601, IPR2018-01602, IPR2018-01603	Splash2	35 U.S.C. § 103(a)	1, 15, 18, 21, and 22
IPR2018-01601, IPR2018-01602, IPR2018-01603	Splash2 and Gaudiot	35 U.S.C. § 103(a)	1, 15, 18, 21, and 22
IPR2018-01601	Splash2 and RaPiD	35 U.S.C. § 103(a)	8 and 9
IPR2018-01601	Splash2, RaPiD, and Gaudiot[5]	35 U.S.C. § 103(a)	8 and 9
IPR2018-01601	Splash2 and Jeong	35 U.S.C. § 103(a)	20
IPR2018-01601	Splash2, Jeong, and Gaudiot	35 U.S.C. § 103(a)	20
IPR2018-01602	Splash2 and Chunky SLD	35 U.S.C. § 103(a)	7, 17, and 24
IPR2018-01602	Splash2, Chunky SLD, and Gaudiot	35 U.S.C. § 103(a)	7, 17, and 24
IPR2018-01603	Splash2 and Roccatano	35 U.S.C. § 103(a)	2–5, 22, and 23
IPR2018-01603	Splash2, Roccatano, and Gaudiot	35 U.S.C. § 103(a)	2–5, 22, and 23

1601 Institution Decision 10-11.

IPR2018-01605: Trial of 800 Patent



Case Number(s)	Reference(s)	Basis	Claim(s) Challenged
IPR2018-01605, IPR2018-01606, IPR2018-01607	Splash2	35 U.S.C. §§ 102(a) and 102(b)[4]	1, 15, 18, 21, and 22
IPR2018-01605, IPR2018-01606, IPR2018-01607	Splash2	35 U.S.C. § 103(a)	1, 15, 18, 21, and 22
IPR2018-01605, IPR2018-01606, IPR2018-01607	Splash2 and Gaudiot	35 U.S.C. § 103(a)	1, 15, 18, 21, and 22
IPR2018-01605	Splash2 and RaPiD	35 U.S.C. § 103(a)	8 and 9
IPR2018-01605	Splash2, RaPiD, and Gaudiot[5]	35 U.S.C. § 103(a)	8 and 9
IPR2018-01605	Splash2 and Jeong	35 U.S.C. § 103(a)	20
IPR2018-01605	Splash2, Jeong, and Gaudiot	35 U.S.C. § 103(a)	20
IPR2018-01606	Splash2 and Chunky SLD	35 U.S.C. § 103(a)	7, 17, and 24
IPR2018-01606	Splash2, Chunky SLD, and Gaudiot	35 U.S.C. § 103(a)	7, 17, and 24
IPR2018-01607	Splash2 and Roccatano	35 U.S.C. § 103(a)	2–5, 22, and 23
IPR2018-01607	Splash2, Roccatano, and Gaudiot	35 U.S.C. § 103(a)	2–5, 22, and 23

1605 Institution Decision 10-11.

Roadmap

324 and 800 Patent Overviews

Prior Art Overview

Patentability Issues

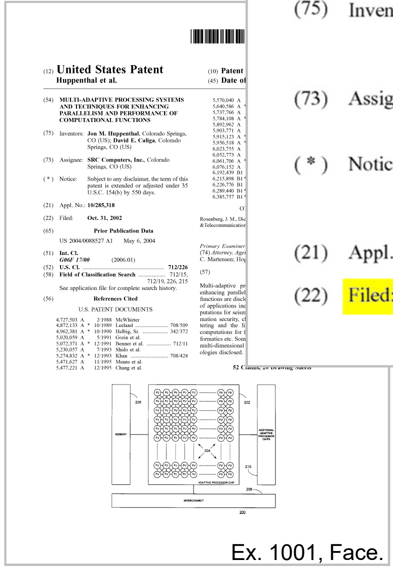
Roadmap

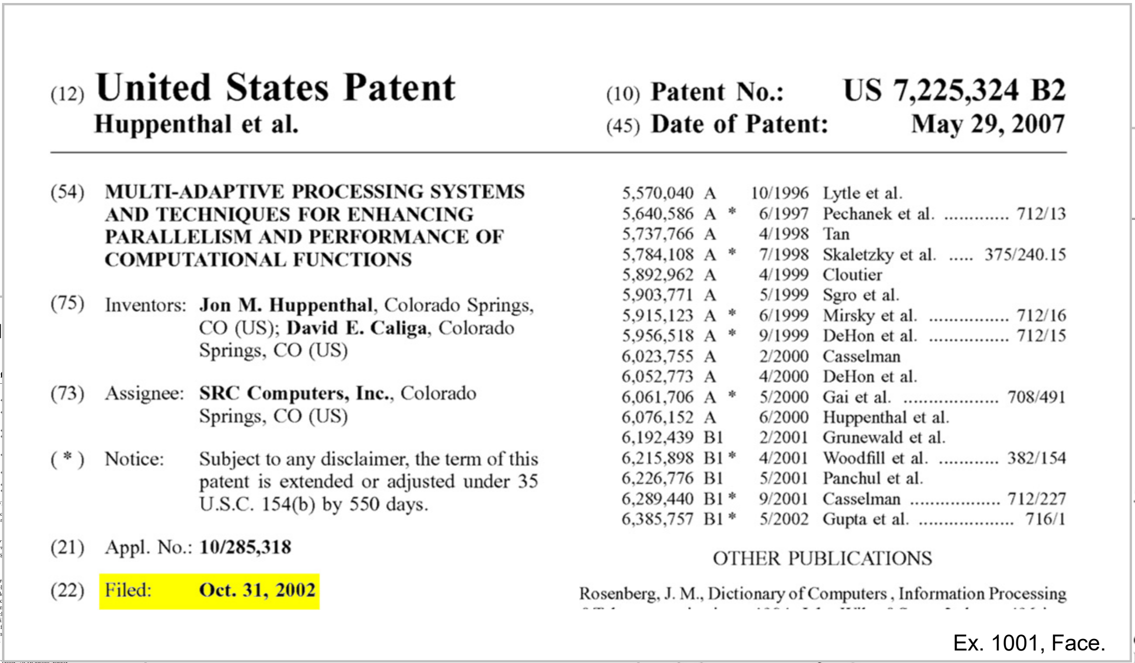
324 and 800 Patent Overviews

Prior Art Overview

Patentability Issues

324/800 Patent Overview


Ex. 1001, Face.


(12) **United States Patent**
Huppenthal et al.

(10) **Patent No.:** **US 7,225,324 B2**
(45) **Date of Patent:** **May 29, 2007**

(54) **MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS**

(75) Inventors: **Jon M. Huppenthal**, Colorado Springs, CO (US); **David E. Caliga**, Colorado Springs, CO (US)

(73) Assignee: **SRC Computers, Inc.**, Colorado Springs, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 550 days.

(21) Appl. No.: **10/285,318**

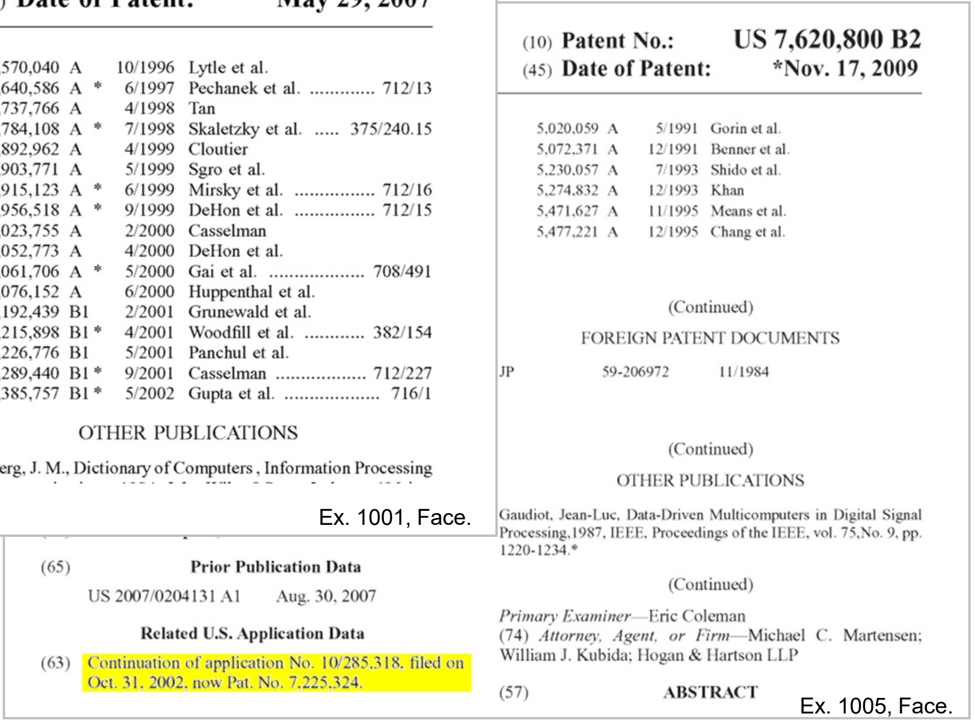
(22) Filed: **Oct. 31, 2002**

5,570,040 A		10/1996	Lytle et al.	
5,640,586 A	*	6/1997	Pechanek et al.	712/13
5,737,766 A		4/1998	Tan	
5,784,108 A	*	7/1998	Skaletzky et al.	375/240.15
5,892,962 A		4/1999	Cloutier	
5,903,771 A		5/1999	Sgro et al.	
5,915,123 A	*	6/1999	Mirsky et al.	712/16
5,956,518 A	*	9/1999	DeHon et al.	712/15
6,023,755 A		2/2000	Casselman	
6,052,773 A		4/2000	DeHon et al.	
6,061,706 A	*	5/2000	Gai et al.	708/491
6,076,152 A		6/2000	Huppenthal et al.	
6,192,439 B1		2/2001	Grunewald et al.	
6,215,898 B1	*	4/2001	Woodfill et al.	382/154
6,226,776 B1		5/2001	Panchul et al.	
6,289,440 B1	*	9/2001	Casselman	712/227
6,385,757 B1	*	5/2002	Gupta et al.	716/1

OTHER PUBLICATIONS

Rosenberg, J. M., Dictionary of Computers, Information Processing

Ex. 1001, Face.


(10) **Patent No.:** **US 7,620,800 B2**
(45) **Date of Patent:** ***Nov. 17, 2009**

5,020,059 A	5/1991	Gorin et al.
5,072,371 A	12/1991	Benner et al.
5,230,057 A	7/1993	Shido et al.
5,274,832 A	12/1993	Khan
5,471,627 A	11/1995	Means et al.
5,477,221 A	12/1995	Chang et al.

(Continued)

FOREIGN PATENT DOCUMENTS

JP	59-206972	11/1984

(Continued)

OTHER PUBLICATIONS

Gaudiot, Jean-Luc, Data-Driven Multicomputers in Digital Signal Processing, 1987, IEEE, Proceedings of the IEEE, vol. 75, No. 9, pp. 1220-1234.*

(Continued)

(65) **Prior Publication Data**

US 2007/0204131 A1 Aug. 30, 2007

Related U.S. Application Data

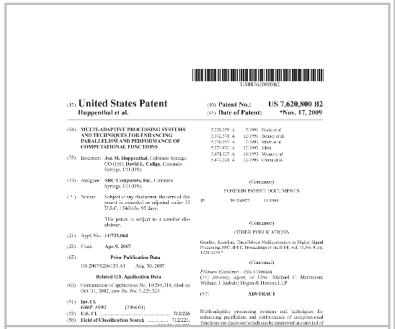
(63) Continuation of application No. 10/285,318, filed on Oct. 31, 2002, now Pat. No. 7,225,324.

Primary Examiner—Eric Coleman
(74) *Attorney, Agent, or Firm*—Michael C. Martensen; William J. Kubida; Hogan & Hartson LLP

(57) **ABSTRACT**

Ex. 1005, Face.

324/800 Patent Overview



US007620800B2

(12) United States Patent
Huppenthal et al.

(10) Patent No.: US 7,620,800 B2
(45) Date of Patent: *Nov. 17, 2009

(54) MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS

(75) Inventors: **Jon M. Huppenthal**, Colorado Springs, CO (US); **David E. Caliga**, Colorado Springs, CO (US)

(73) Assignee: **SRC Computers, Inc.**, Colorado Springs, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 95 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/733,064**

(22) Filed: **Apr. 9, 2007**

(65) **Prior Publication Data**

US 2007/0204131 A1 Aug. 30, 2007

Related U.S. Application Data

(63) Continuation of application No. 10/285,318, filed on Oct. 31, 2002, now Pat. No. 7,225,324.

(51) **Int. Cl.**
G06F 15/82 (2006.01)

(52) **U.S. Cl.** **712/226**

(58) **Field of Classification Search** 712/226,

5,020,059 A	5/1991	Gorin et al.	
5,072,371 A	12/1991	Benner et al.	
5,230,057 A	7/1993	Shido et al.	
5,274,832 A	12/1993	Khan	
5,471,627 A	11/1995	Means et al.	
5,477,221 A	12/1995	Chang et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

JP 59-206972 11/1984

(Continued)

OTHER PUBLICATIONS

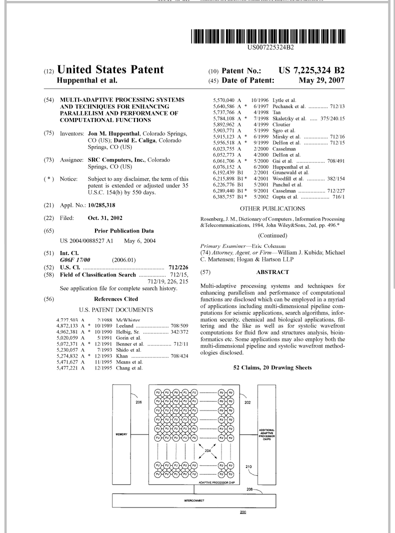
Gaudiot, Jean-Luc, Data-Driven Multicomputers in Digital Signal Processing,1987, IEEE, Proceedings of the IEEE, vol. 75, No. 9, pp. 1220-1234.*

(Continued)

Primary Examiner—Eric Coleman
(74) *Attorney, Agent, or Firm*—Michael C. Martensen; William J. Kubida; Hogan & Hartson LLP

(57) **ABSTRACT**

Multi-adaptive processing systems and techniques for enhancing parallelism and performance of computational functions are disclosed which can be employed in a myriad of



US007225324B2

(12) **United States Patent**
Huppenthal et al.

(10) **Patent No.: US 7,225,324 B2**
(45) **Date of Patent: May 29, 2007**

(54) MULTI-ADAPTIVE PROCESSING SYSTEMS AND TECHNIQUES FOR ENHANCING PARALLELISM AND PERFORMANCE OF COMPUTATIONAL FUNCTIONS

(75) Inventors: **Jon M. Huppenthal**, Colorado Springs, CO (US); **David E. Caliga**, Colorado Springs, CO (US)

(73) Assignee: **SRC Computers, Inc.**, Colorado Springs, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 550 days.

(21) Appl. No.: **10/285,318**

(22) Filed: **Oct. 31, 2002**

(65) **Prior Publication Data**

US 2004/0088527 A1 May 6, 2004

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **712/226**

(58) **Field of Classification Search** 712/15, 712/19, 226, 215

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,727,503 A		2/1988	McWhirter	
4,872,133 A	*	10/1989	Leeland	708/509
4,962,381 A	*	10/1990	Helbig, Sr.	342/372
5,020,059 A		5/1991	Gorin et al.	
5,072,371 A	*	12/1991	Benner et al.	712/11
5,230,057 A		7/1993	Shido et al.	
5,274,832 A	*	12/1993	Khan	708/424
5,471,627 A		11/1995	Means et al.	
5,477,221 A		12/1995	Chang et al.	
5,570,040 A		10/1996	Lytle et al.	
5,640,586 A	*	6/1997	Pechanek et al.	712/13
5,737,766 A		4/1998	Tan	
5,784,108 A	*	7/1998	Skaletzky et al.	375/240.15
5,892,962 A		4/1999	Cloutier	
5,903,771 A		5/1999	Sgro et al.	
5,915,123 A	*	6/1999	Mirsky et al.	712/16
5,956,518 A	*	9/1999	DeHon et al.	712/15
6,023,755 A		2/2000	Casselman	
6,052,773 A		4/2000	DeHon et al.	
6,061,706 A	*	5/2000	Gai et al.	708/491
6,076,152 A		6/2000	Huppenthal et al.	
6,192,439 B1		2/2001	Grunewald et al.	
6,215,898 B1	*	4/2001	Woodfill et al.	382/154
6,226,776 B1		5/2001	Panchul et al.	
6,289,440 B1	*	9/2001	Casselman	712/227
6,385,757 B1	*	5/2002	Gupta et al.	716/1

OTHER PUBLICATIONS

Rosenberg, J. M., Dictionary of Computers , Information Processing &Telecommunications, 1984, John Wiley&Sons, 2ed, pp. 496.*

(Continued)

Primary Examiner—Eric Coleman
(74) *Attorney, Agent, or Firm*—William J. Kubida; Michael C. Martensen; Hogan & Hartson LLP

(57) **ABSTRACT**

Multi-adaptive processing systems and techniques for enhancing parallelism and performance of computational functions are disclosed which can be employed in a myriad of applications including multi-dimensional pipeline computations for seismic applications, search algorithms, information security, chemical and biological applications, filtering and the like as well as for systolic wavefront computations for fluid flow and structures analysis, bioinformatics etc. Some applications may also employ both the multi-dimensional pipeline and systolic wavefront methodologies disclosed.

52 Claims, 20 Drawing Sheets

Ex. 1001, Face; Ex. 1005, Face.

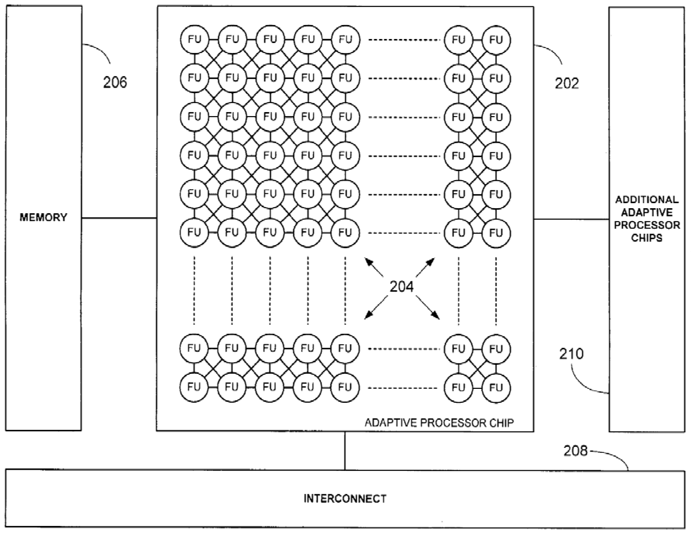
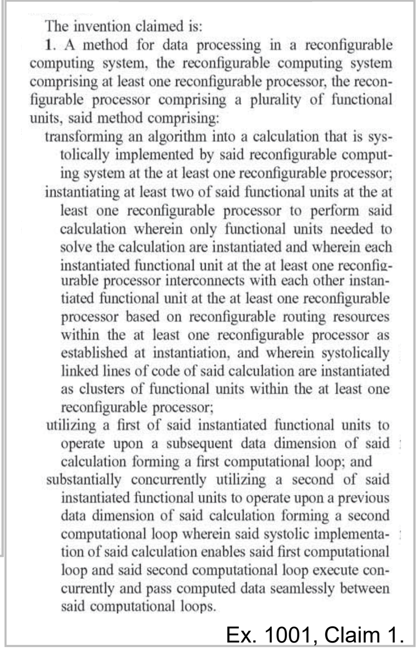


Fig. 2

Ex. 1001, Fig. 2; Ex. 1005, Fig. 2.

324 Patent



The invention claimed is:

1. A method for data processing in a reconfigurable computing system, the reconfigurable computing system comprising at least one reconfigurable processor, the reconfigurable processor comprising a plurality of functional units, said method comprising:

transforming an algorithm into a calculation that is systolically implemented by said reconfigurable computing system at the at least one reconfigurable processor;

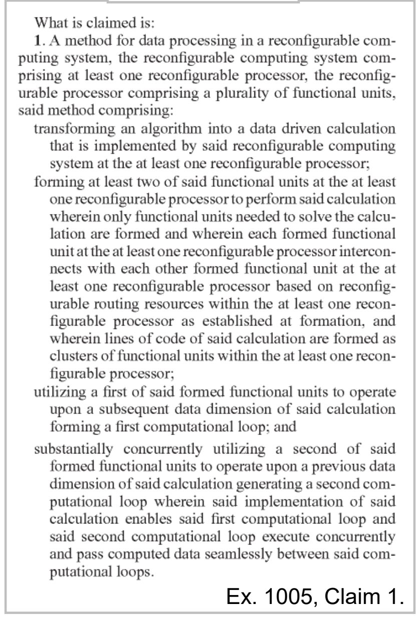
instantiating at least two of said functional units at the at least one reconfigurable processor to perform said calculation wherein only functional units needed to solve the calculation are instantiated and wherein each instantiated functional unit at the at least one reconfigurable processor interconnects with each other instantiated functional unit at the at least one reconfigurable processor based on reconfigurable routing resources within the at least one reconfigurable processor as established at instantiation, and wherein systolically linked lines of code of said calculation are instantiated as clusters of functional units within the at least one reconfigurable processor;

utilizing a first of said instantiated functional units to operate upon a subsequent data dimension of said calculation forming a first computational loop; and

substantially concurrently utilizing a second of said instantiated functional units to operate upon a previous data dimension of said calculation forming a second computational loop wherein said systolic implementation of said calculation enables said first computational loop and said second computational loop execute concurrently and pass computed data seamlessly between said computational loops.

Ex. 1001, Claim 1.

800 Patent



What is claimed is:

1. A method for data processing in a reconfigurable computing system, the reconfigurable computing system comprising at least one reconfigurable processor, the reconfigurable processor comprising a plurality of functional units, said method comprising:

transforming an algorithm into a data driven calculation that is implemented by said reconfigurable computing system at the at least one reconfigurable processor;

forming at least two of said functional units at the at least one reconfigurable processor to perform said calculation wherein only functional units needed to solve the calculation are formed and wherein each formed functional unit at the at least one reconfigurable processor interconnects with each other formed functional unit at the at least one reconfigurable processor based on reconfigurable routing resources within the at least one reconfigurable processor as established at formation, and wherein lines of code of said calculation are formed as clusters of functional units within the at least one reconfigurable processor;

utilizing a first of said formed functional units to operate upon a subsequent data dimension of said calculation forming a first computational loop; and

substantially concurrently utilizing a second of said formed functional units to operate upon a previous data dimension of said calculation generating a second computational loop wherein said implementation of said calculation enables said first computational loop and said second computational loop execute concurrently and pass computed data seamlessly between said computational loops.

Ex. 1005, Claim 1.

Roadmap

324 and 800 Patent Overviews

Prior Art Overview

Patentability Issues

Splash2: FPGAs in a Custom Computing Machine (1996)

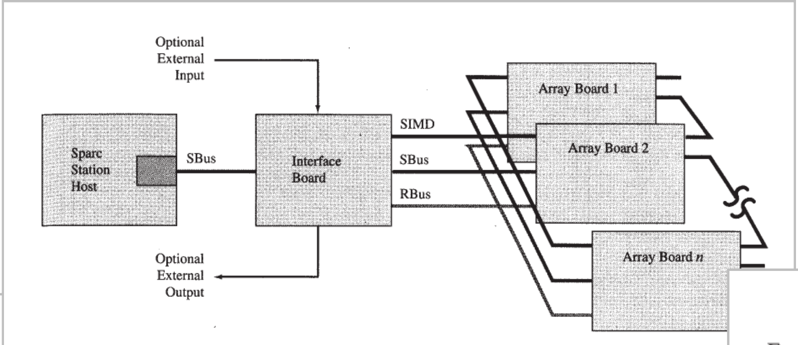
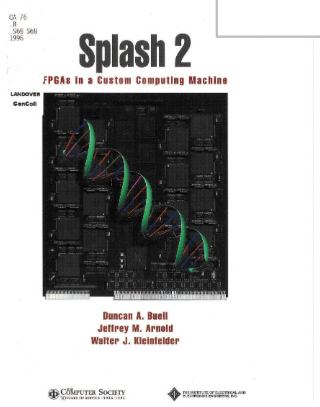


FIGURE 2.3 Splash 2 System Architecture

Ex. 1007, Fig. 2.3(cited in 1601 Pet., 22).



Ex. 1007, Face.

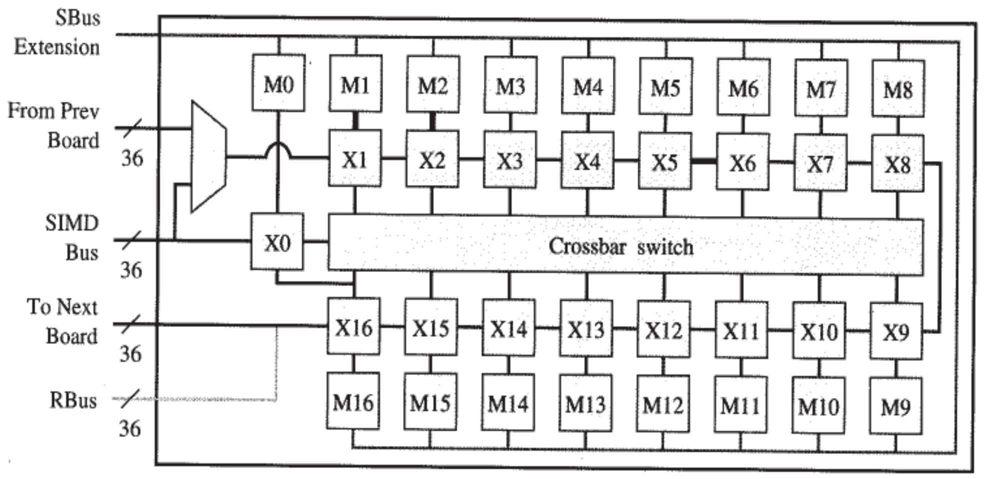
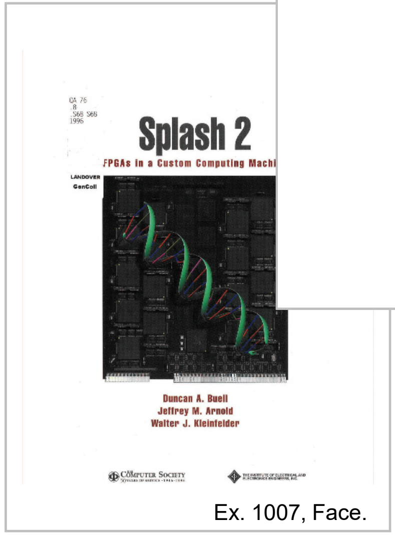


FIGURE 2.4 Array Board Architecture

Ex. 1007, Fig. 2.4 (cited in 1601 Pet., 23)

Splash2



QA 76
.8
.S68 S68
1996

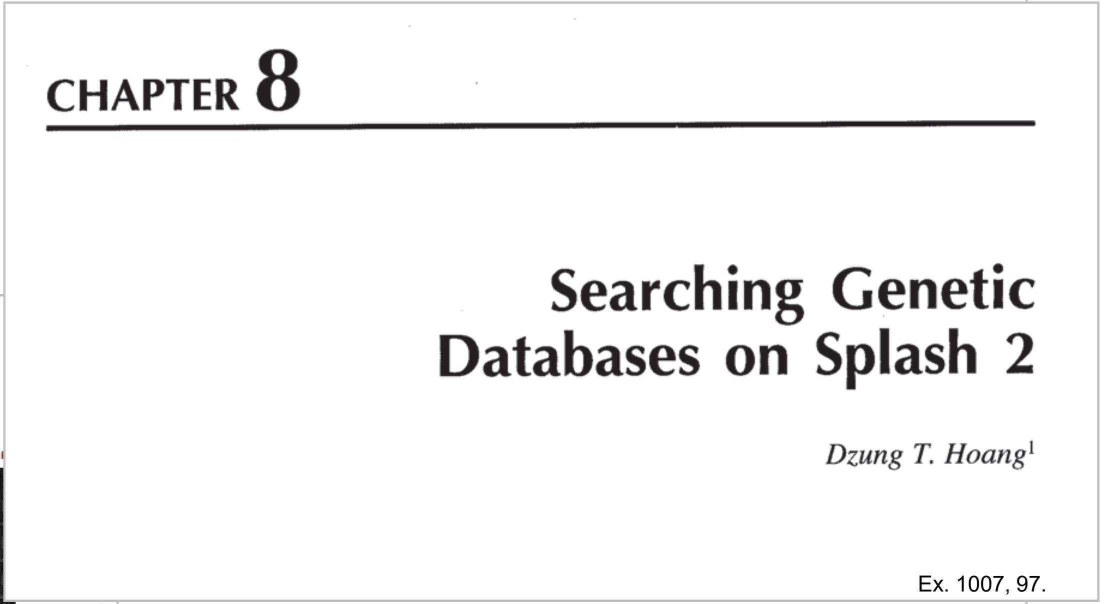
Splash 2

FPGAs in a Custom Computing Machi

LANDOVER
GenColl

Duncan A. Buell
Jeffrey M. Arnold
Walter J. Kleinfelder

Ex. 1007, Face.

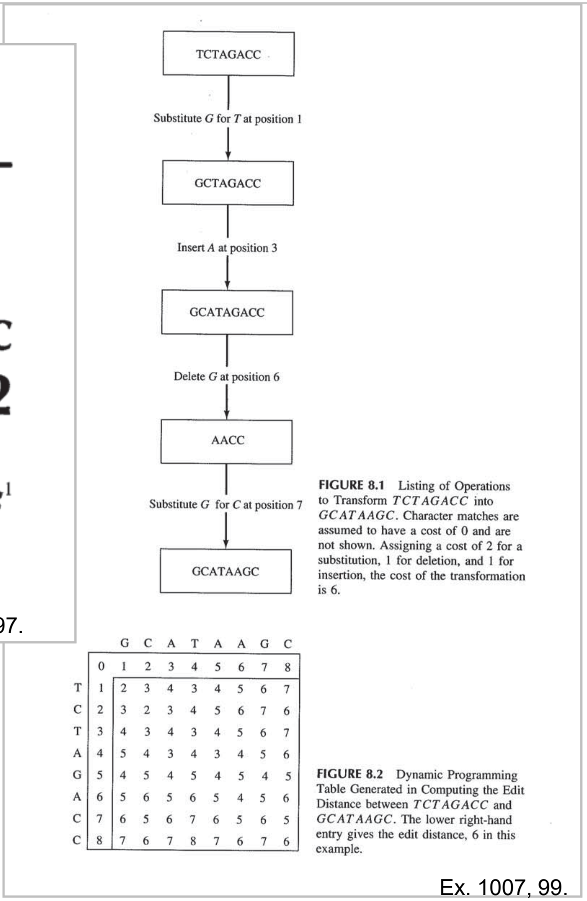


CHAPTER 8

Searching Genetic Databases on Splash 2

Dzung T. Hoang[1]

Ex. 1007, 97.



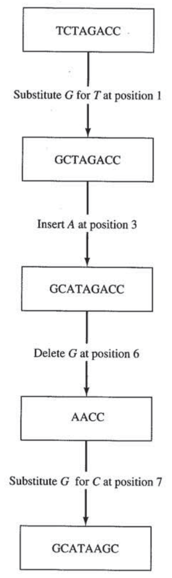


FIGURE 8.1 Listing of Operations to Transform *TCTAGACC* into *GCATAAGC*. Character matches are assumed to have a cost of 0 and are not shown. Assigning a cost of 2 for a substitution, 1 for deletion, and 1 for insertion, the cost of the transformation is 6.

		G	C	A	T	A	A	G	C
	0	1	2	3	4	5	6	7	8
T	1	2	3	4	3	4	5	6	7
C	2	3	2	3	4	5	6	7	6
T	3	4	3	4	3	4	5	6	7
A	4	5	4	3	4	3	4	5	6
G	5	4	5	4	5	4	5	4	5
A	6	5	6	5	6	5	4	5	6
C	7	6	5	6	7	6	5	6	5
C	8	7	6	7	8	7	6	7	6

FIGURE 8.2 Dynamic Programming Table Generated in Computing the Edit Distance between *TCTAGACC* and *GCATAAGC*. The lower right-hand entry gives the edit distance, 6 in this example.

Ex. 1007, 99.

Petitioner's Demonstrative – Not Evidence

Splash2 (Chapter 8) – Unidirectional Array

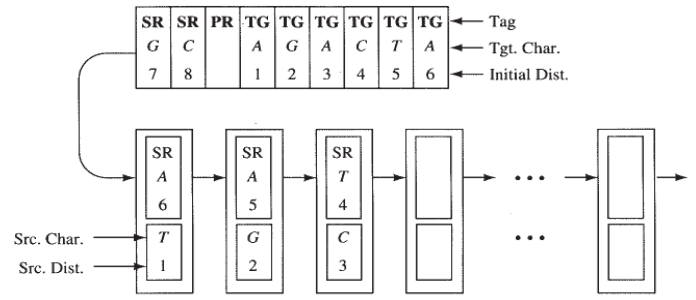


FIGURE 8.9 Data Flow through the Unidirectional Systolic Array. The source sequence is first loaded into the array. The target sequences are then streamed through the array. The tag acts as a simple instruction telling each PE how to process the incoming data. The SR tag instructs an empty PE to load the source character and distance from the input stream. The PR tag marks the end of the source stream. The TG tag signals a target character. Multiple source and target sequences can be carried on the input stream for uninterrupted pipelined processing.

Ex. 1007, Fig. 8.9 (cited in 1601 Pet., 27).

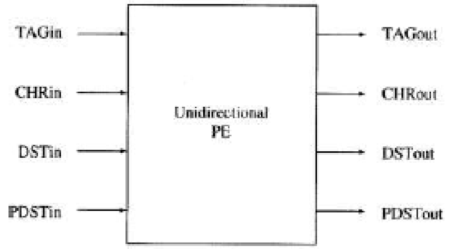
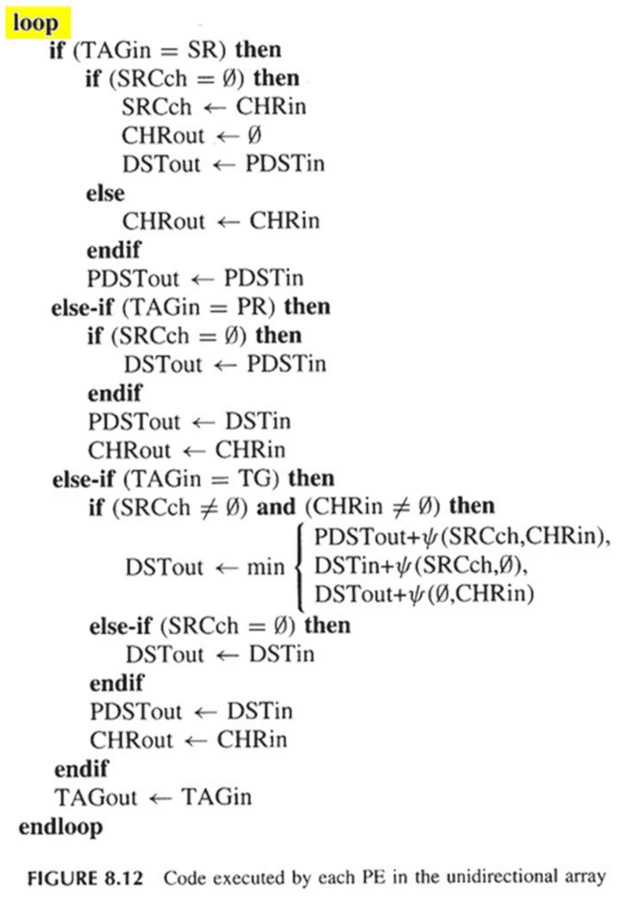


FIGURE 8.10 Processing Element for Unidirectional Array

Ex. 1007, Fig. 8.10 (cited in 1601 Pet., 28).



```
loop
  if (TAGin = SR) then
    if (SRCch = Ø) then
      SRCch ← CHRin
      CHRout ← Ø
      DSTout ← PDSTin
    else
      CHRout ← CHRin
    endif
    PDSTout ← PDSTin
  else-if (TAGin = PR) then
    if (SRCch = Ø) then
      DSTout ← PDSTin
    endif
    PDSTout ← DSTin
    CHRout ← CHRin
  else-if (TAGin = TG) then
    if (SRCch ≠ Ø) and (CHRin ≠ Ø) then
                      ⎧ PDSTout+ψ(SRCch,CHRin),
      DSTout ← min    ⎨ DSTin+ψ(SRCch,Ø),
                      ⎩ DSTout+ψ(Ø,CHRin)
    else-if (SRCch = Ø) then
      DSTout ← DSTin
    endif
    PDSTout ← DSTin
    CHRout ← CHRin
  endif
  TAGout ← TAGin
endloop
```

FIGURE 8.12 Code executed by each PE in the unidirectional array

Ex. 1007, Fig. 8.12 (cited in 1601 Pet., 28).

Splash2 (Chapter 8) – Bidirectional Array

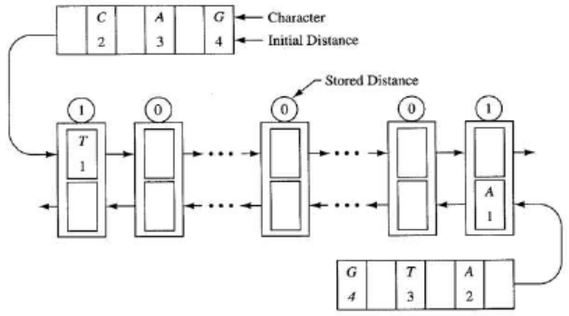


FIGURE 8.5 Data Flow through the Bidirectional Systolic Array. The source and target sequences are streamed through the array in opposite directions. A comparison is performed when a source character and a target character meet in a PE.

Ex. 1007, Fig. 8.5 (cited in 1601 Pet., 25).

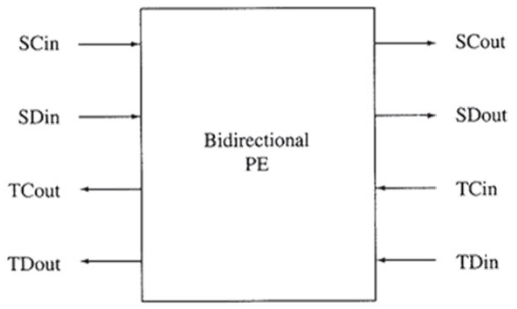


FIGURE 8.6 Processing Element for Bidirectional Array

```
loop
    if (SCin ≠ Ø) and (TCin ≠ Ø) then
        PEDist ← min { PEDist+ψ(SCin,TCin),
                       TDin+ψ(SCin,Ø),
                       SDin+ψ(Ø,TCin) }
    else-if (SCin ≠ Ø) then
        PEDist ← SDin
    else-if (TCin ≠ Ø) then
        PEDist ← TDin
    endif
    SCout ← SCin
    TCout ← TCin
    SDout ← PEDist
    TDout ← PEDist
endloop
```

FIGURE 8.7 Code Executed by Each PE in the Bidirectional Array

Ex. 1007, Fig. 8.6, Fig. 8.7 (cited in 1601 Pet., 26)

Secondary References

Gaudiot

Sep. 1987
(Data-driven processing techniques)

RaPiD

Apr. 1997
(Systolic DCT calculations)

Jeong

June 1997
(Systolic encryption calculations)

ChunkySLD

Apr. 1997
(Systolic target resolution calculations)

Roccatano

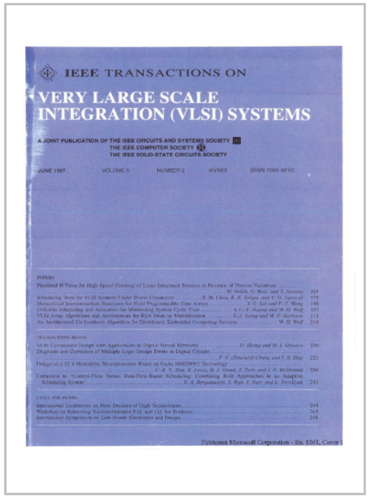
May 1998
(Systolic molecular dynamics calculations)

proceedings OF THE IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS september 1987
SPECIAL ISSUE ON
hardware and software for DSP

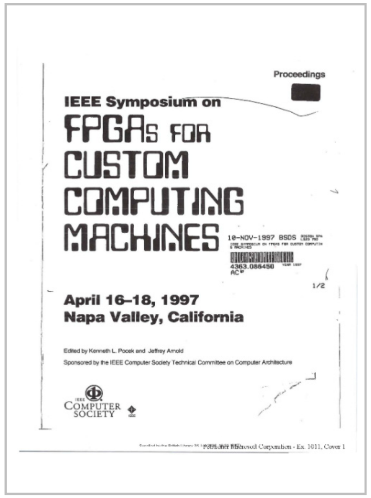
Ex. 1010 (cited in 1601 Pet., 52 et seq).

Proceedings
IEEE Symposium on
FPGAs FOR CUSTOM COMPUTING MACHINES
April 16–18, 1997
Napa Valley, California
Edited by Kenneth L. Pocek and Jeffrey Arnold
Sponsored by the IEEE Computer Society Technical Committee on Computer Architecture
IEEE COMPUTER SOCIETY

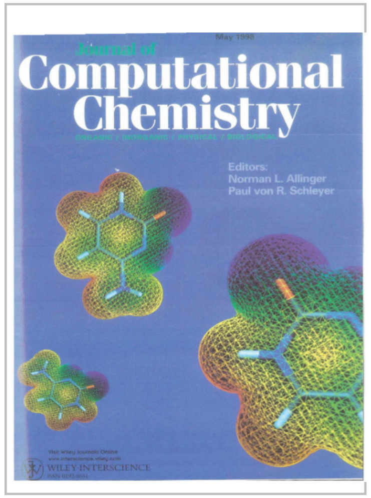
Ex. 1009 (cited in 1601 Pet., 55 et seq).


IEEE TRANSACTIONS ON
VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS
A JOINT PUBLICATION OF THE IEEE CIRCUITS AND SYSTEMS SOCIETY
THE IEEE COMPUTER SOCIETY
THE IEEE SOLID-STATE CIRCUITS SOCIETY
JUNE 1997 VOLUME 5 NUMBER 2
Petitioner Microsoft Corporation - Ex. 1061, Cover 1

Ex. 1061 (cited in 1601 Pet., 67 et seq).


Proceedings
IEEE Symposium on
FPGAs FOR CUSTOM COMPUTING MACHINES
April 16–18, 1997
Napa Valley, California
Edited by Kenneth L. Pocek and Jeffrey Arnold
Sponsored by the IEEE Computer Society Technical Committee on Computer Architecture
IEEE COMPUTER SOCIETY
Petitioner Microsoft Corporation - Ex. 1011, Cover 1

Ex. 1011 (cited in 1602 Pet., 63 et seq).


May 1998
Journal of
Computational Chemistry
Editors:
Norman L. Allinger
Paul von R. Schleyer
WILEY-INTERSCIENCE

Ex. 1012 (cited in 1603 Pet., 67 et seq).

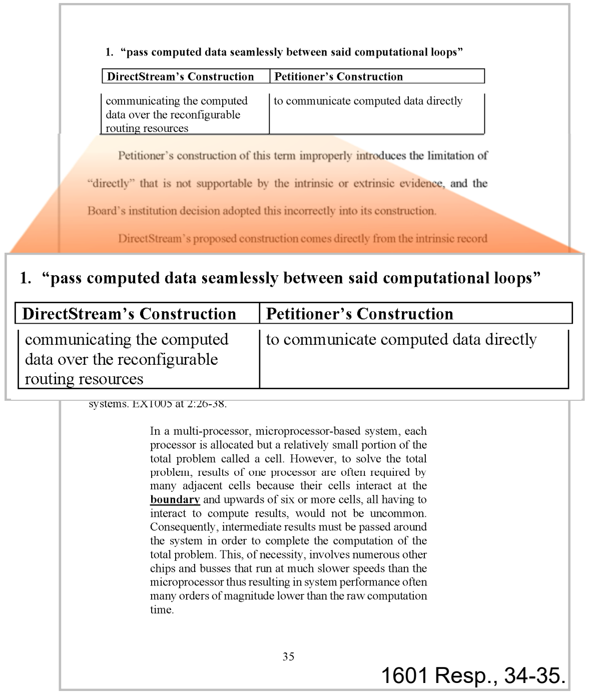
Roadmap

324 and 800 Patent Overviews

Prior Art Overview

Patentability Issues

Claim Construction – Seamlessly Passing Data



1. **"pass computed data seamlessly between said computational loops"**

DirectStream's Construction	**Petitioner's Construction**
communicating the computed data over the reconfigurable routing resources	to communicate computed data directly

Petitioner's construction of this term improperly introduces the limitation of "directly" that is not supportable by the intrinsic or extrinsic evidence, and the Board's institution decision adopted this incorrectly into its construction.

DirectStream's proposed construction comes directly from the intrinsic record

1. **"pass computed data seamlessly between said computational loops"**

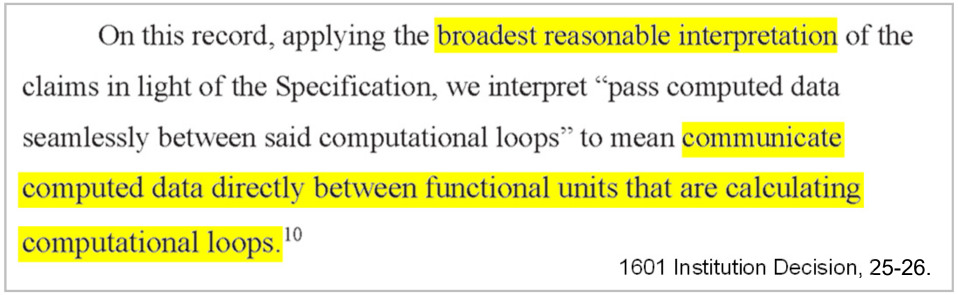
DirectStream's Construction	**Petitioner's Construction**
communicating the computed data over the reconfigurable routing resources	to communicate computed data directly

systems. EX1005 at 2:26-38.

> In a multi-processor, microprocessor-based system, each processor is allocated but a relatively small portion of the total problem called a cell. However, to solve the total problem, results of one processor are often required by many adjacent cells because their cells interact at the **boundary** and upwards of six or more cells, all having to interact to compute results, would not be uncommon. Consequently, intermediate results must be passed around the system in order to complete the computation of the total problem. This, of necessity, involves numerous other chips and busses that run at much slower speeds than the microprocessor thus resulting in system performance often many orders of magnitude lower than the raw computation time.

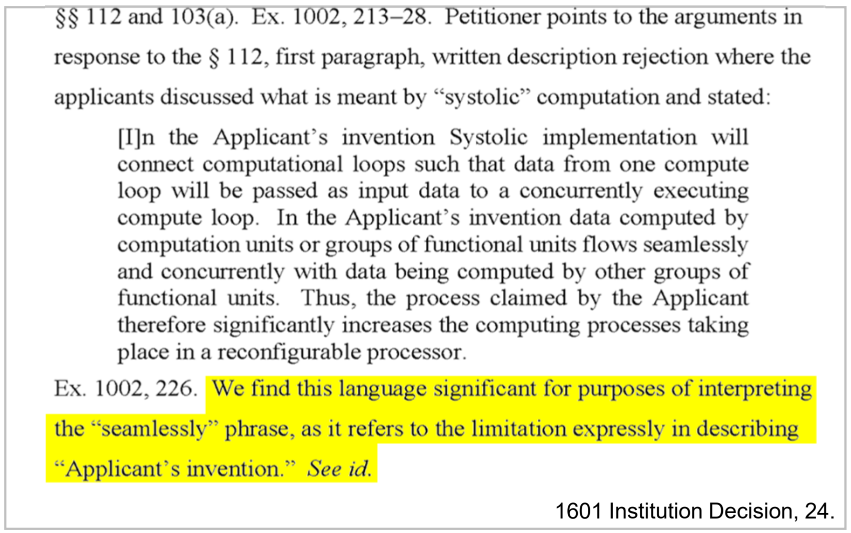
35

1601 Resp., 34-35.



On this record, applying the broadest reasonable interpretation of the claims in light of the Specification, we interpret "pass computed data seamlessly between said computational loops" to mean communicate computed data directly between functional units that are calculating computational loops.[10]

1601 Institution Decision, 25-26.



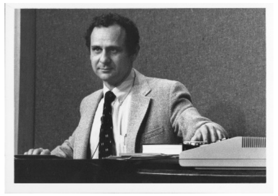
§§ 112 and 103(a). Ex. 1002, 213–28. Petitioner points to the arguments in response to the § 112, first paragraph, written description rejection where the applicants discussed what is meant by "systolic" computation and stated:

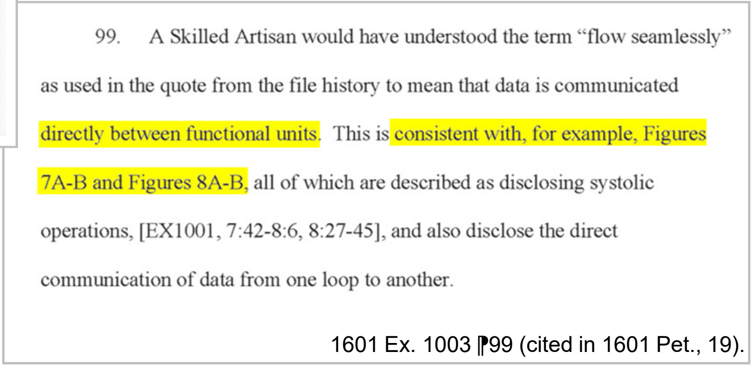
> [I]n the Applicant's invention Systolic implementation will connect computational loops such that data from one compute loop will be passed as input data to a concurrently executing compute loop. In the Applicant's invention data computed by computation units or groups of functional units flows seamlessly and concurrently with data being computed by other groups of functional units. Thus, the process claimed by the Applicant therefore significantly increases the computing processes taking place in a reconfigurable processor.

Ex. 1002, 226. We find this language significant for purposes of interpreting the "seamlessly" phrase, as it refers to the limitation expressly in describing "Applicant's invention." *See id.*

1601 Institution Decision, 24.

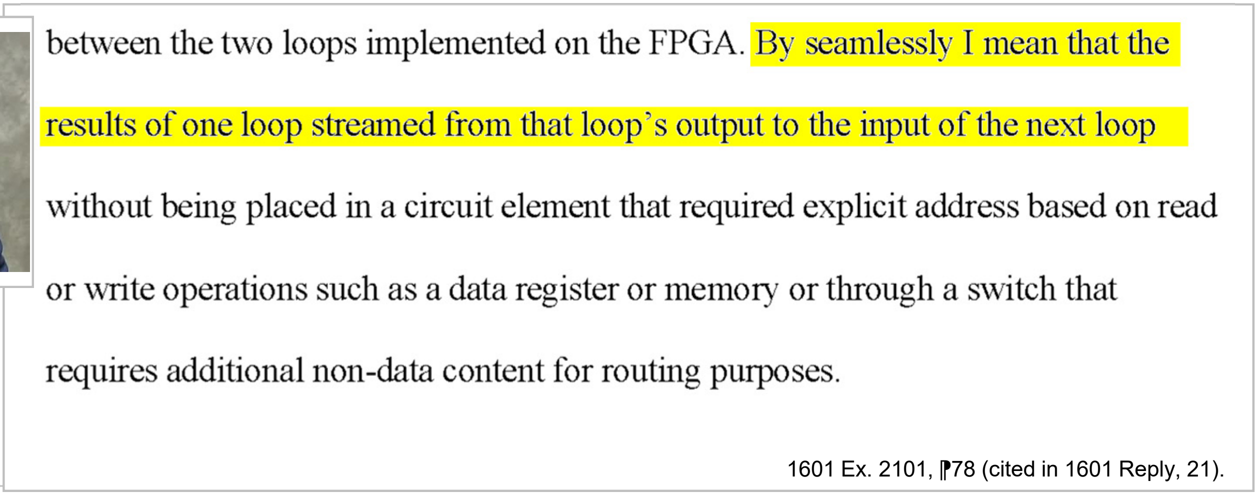
Claim Construction – Seamlessly Passing Data





99. A Skilled Artisan would have understood the term "flow seamlessly" as used in the quote from the file history to mean that data is communicated directly between functional units. This is consistent with, for example, Figures 7A-B and Figures 8A-B, all of which are described as disclosing systolic operations, [EX1001, 7:42-8:6, 8:27-45], and also disclose the direct communication of data from one loop to another.

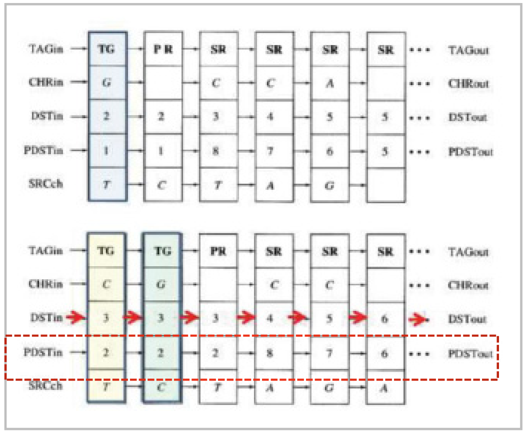
1601 Ex. 1003 ⁋99 (cited in 1601 Pet., 19).



between the two loops implemented on the FPGA. By seamlessly I mean that the results of one loop streamed from that loop's output to the input of the next loop without being placed in a circuit element that required explicit address based on read or write operations such as a data register or memory or through a switch that requires additional non-data content for routing purposes.

1601 Ex. 2101, ⁋78 (cited in 1601 Reply, 21).

Splash2 - Seamlessly Passing Data



277. Figure 8.13, reproduced below with annotations, further shows the seamless communication of the DST data in the systolic calculation as a sequence of red arrows. The red arrows indicate, for example, that the computed output DSTout from a functional unit is directly connected to the next functional unit input DSTin. Similarly, all output data from one functional unit are seamlessly connected to the corresponding inputs of the functional unit in the sequence. There are no intermediate interfaces between modules to translate the output of one module into a form where it can be used as input data to the next module. Such interfaces, if they were to exist, would be "seams" in the communication links between adjacent functional units.

324 Ex. 1003 ¶277 (cited in 1601 Pet., 46).

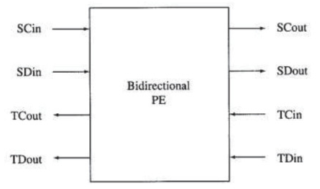


FIGURE 8.6 Processing Element for Bidirectional Array

Id., 101.

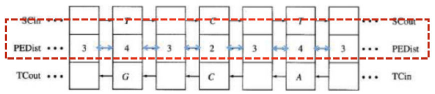
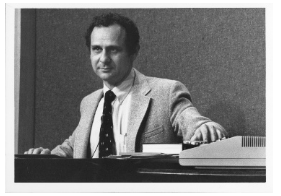


FIGURE 8.8 Trace of Bidirectional Array When Comparing the Sequences *TCTAGACC* and *GCATAAGC*

278. Figures 8.6 and 8.8 disclose the same functionality for the Bidirectional implementations: The blue arrows indicate that the computed output PEDist from a functional unit is directly connected to the next functional unit input PEDist in both directions. The code for the Bidirectional implementation discloses that the computed PEDist is conveyed to the left on output TDout, and to the right on output SDout. EX1007, 101.

1601 Ex. 1003 ¶278 (cited in 1601 Pet., 46).



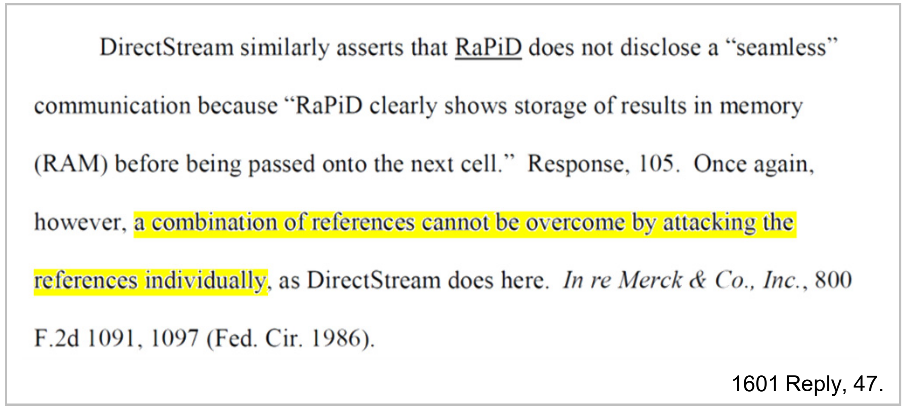
A. I -- I'm puzzled because that -- that register would be within -- within the processing element in my mind.

Q. Okay.

A. If it's within the processing element as a register, yeah, I would put it there, then the output of that register, if it's connected directly to the input of the next processing element, would be direct.

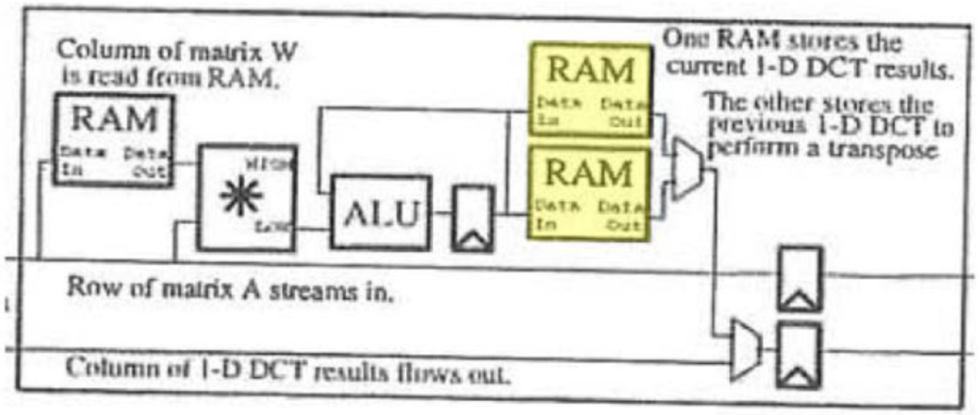
Ex. 2064, 86:21-87:5 (cited in 1601 Reply, 25).

Splash2/RaPiD - Seamlessly Passing Data



DirectStream similarly asserts that RaPiD does not disclose a "seamless" communication because "RaPiD clearly shows storage of results in memory (RAM) before being passed onto the next cell." Response, 105. Once again, however, a combination of references cannot be overcome by attacking the references individually, as DirectStream does here. *In re Merck & Co., Inc.*, 800 F.2d 1091, 1097 (Fed. Cir. 1986).

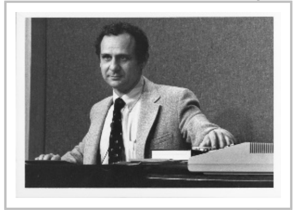
1601 Reply, 47.

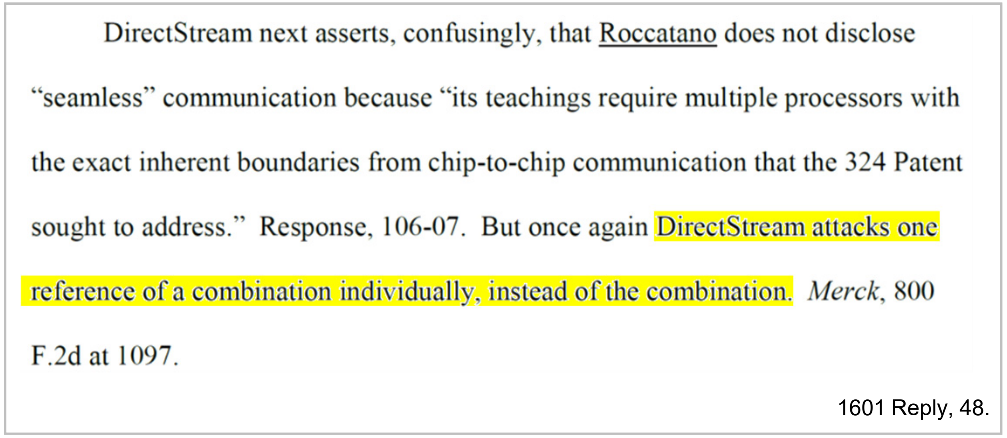


Column of matrix W is read from RAM.
RAM
Data In
Data Out
High
Low
ALU
RAM
Data In
Data Out
RAM
Data In
Data Out
One RAM stores the current 1-D DCT results.
The other stores the previous 1-D DCT to perform a transpose
Row of matrix A streams in.
Column of 1-D DCT results flows out.

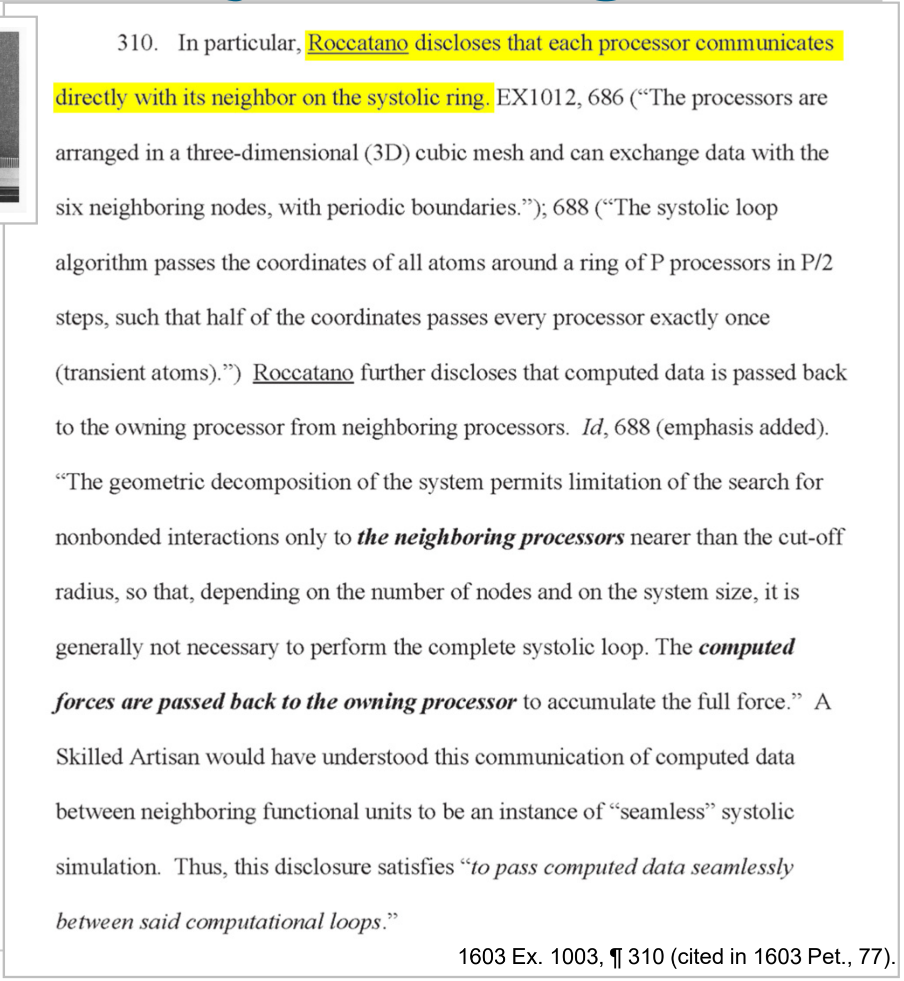
1601 Resp., 106.

Splash2/Roccatano - Seamlessly Passing Data



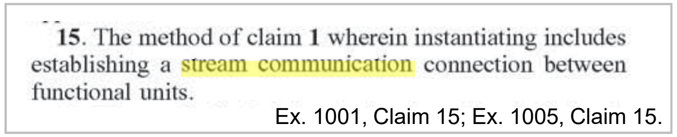

DirectStream next asserts, confusingly, that Roccatano does not disclose "seamless" communication because "its teachings require multiple processors with the exact inherent boundaries from chip-to-chip communication that the 324 Patent sought to address." Response, 106-07. But once again DirectStream attacks one reference of a combination individually, instead of the combination. *Merck*, 800 F.2d at 1097.

1601 Reply, 48.

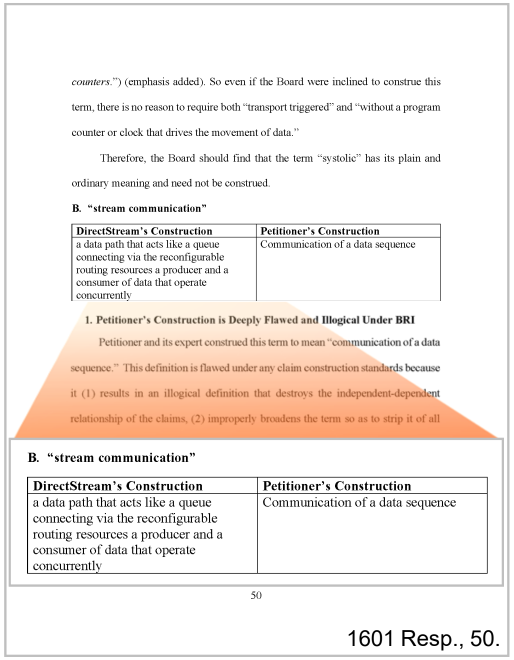

310. In particular, Roccatano discloses that each processor communicates directly with its neighbor on the systolic ring. EX1012, 686 ("The processors are arranged in a three-dimensional (3D) cubic mesh and can exchange data with the six neighboring nodes, with periodic boundaries."); 688 ("The systolic loop algorithm passes the coordinates of all atoms around a ring of P processors in P/2 steps, such that half of the coordinates passes every processor exactly once (transient atoms).") Roccatano further discloses that computed data is passed back to the owning processor from neighboring processors. *Id*, 688 (emphasis added). "The geometric decomposition of the system permits limitation of the search for nonbonded interactions only to ***the neighboring processors*** nearer than the cut-off radius, so that, depending on the number of nodes and on the system size, it is generally not necessary to perform the complete systolic loop. The ***computed forces are passed back to the owning processor*** to accumulate the full force." A Skilled Artisan would have understood this communication of computed data between neighboring functional units to be an instance of "seamless" systolic simulation. Thus, this disclosure satisfies "*to pass computed data seamlessly between said computational loops*."

1603 Ex. 1003, ¶ 310 (cited in 1603 Pet., 77).

Claim Construction – Stream Communication


15. The method of claim 1 wherein instantiating includes establishing a stream communication connection between functional units.

Ex. 1001, Claim 15; Ex. 1005, Claim 15.


counters.") (emphasis added). So even if the Board were inclined to construe this term, there is no reason to require both "transport triggered" and "without a program counter or clock that drives the movement of data."

Therefore, the Board should find that the term "systolic" has its plain and ordinary meaning and need not be construed.

B. "stream communication"

DirectStream's Construction	Petitioner's Construction
a data path that acts like a queue connecting via the reconfigurable routing resources a producer and a consumer of data that operate concurrently	Communication of a data sequence

1. Petitioner's Construction is Deeply Flawed and Illogical Under BRI

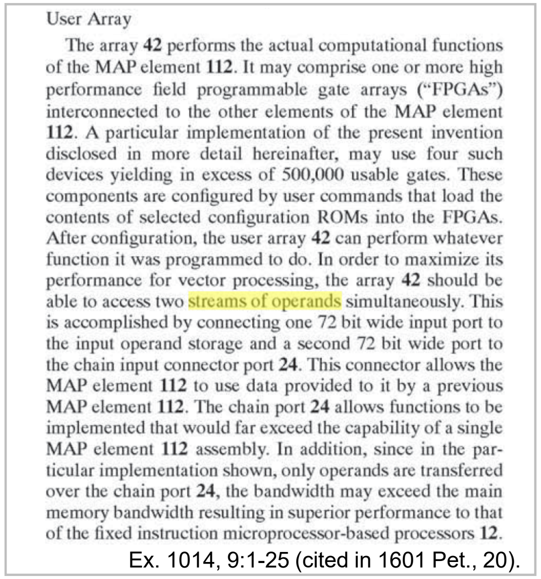
Petitioner and its expert construed this term to mean "communication of a data sequence." This definition is flawed under any claim construction standards because it (1) results in an illogical definition that destroys the independent-dependent relationship of the claims, (2) improperly broadens the term so as to strip it of all

B. "stream communication"

DirectStream's Construction	Petitioner's Construction
a data path that acts like a queue connecting via the reconfigurable routing resources a producer and a consumer of data that operate concurrently	Communication of a data sequence

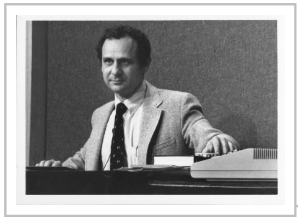
50

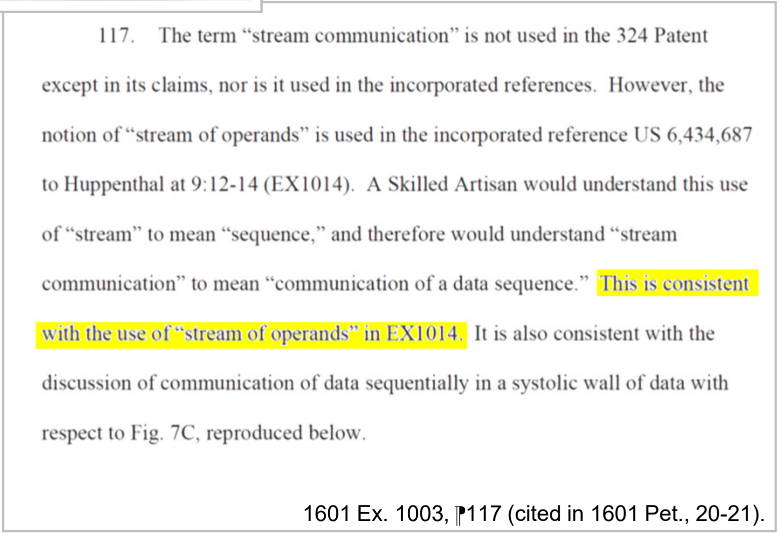
1601 Resp., 50.


User Array

The array **42** performs the actual computational functions of the MAP element **112**. It may comprise one or more high performance field programmable gate arrays ("FPGAs") interconnected to the other elements of the MAP element **112**. A particular implementation of the present invention disclosed in more detail hereinafter, may use four such devices yielding in excess of 500,000 usable gates. These components are configured by user commands that load the contents of selected configuration ROMs into the FPGAs. After configuration, the user array **42** can perform whatever function it was programmed to do. In order to maximize its performance for vector processing, the array **42** should be able to access two streams of operands simultaneously. This is accomplished by connecting one 72 bit wide input port to the input operand storage and a second 72 bit wide port to the chain input connector port **24**. This connector allows the MAP element **112** to use data provided to it by a previous MAP element **112**. The chain port **24** allows functions to be implemented that would far exceed the capability of a single MAP element **112** assembly. In addition, since in the particular implementation shown, only operands are transferred over the chain port **24**, the bandwidth may exceed the main memory bandwidth resulting in superior performance to that of the fixed instruction microprocessor-based processors **12**.

Ex. 1014, 9:1-25 (cited in 1601 Pet., 20).

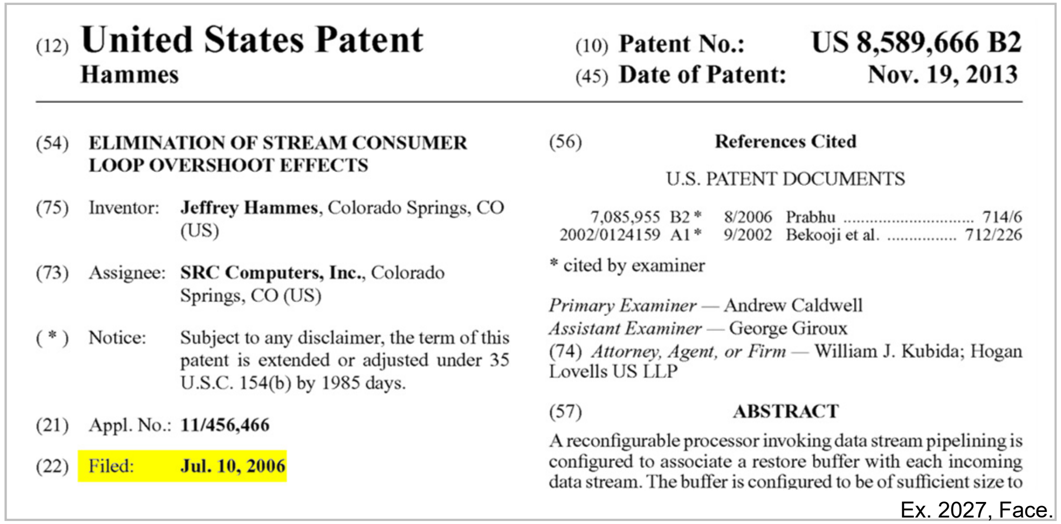



117. The term "stream communication" is not used in the 324 Patent except in its claims, nor is it used in the incorporated references. However, the notion of "stream of operands" is used in the incorporated reference US 6,434,687 to Huppenthal at 9:12-14 (EX1014). A Skilled Artisan would understand this use of "stream" to mean "sequence," and therefore would understand "stream communication" to mean "communication of a data sequence." This is consistent with the use of "stream of operands" in EX1014. It is also consistent with the discussion of communication of data sequentially in a systolic wall of data with respect to Fig. 7C, reproduced below.

1601 Ex. 1003, ⁋117 (cited in 1601 Pet., 20-21).

Claim Construction – Stream Communication

Patent Owner's Proposed Interpretation: "a data path that acts like a queue connecting via the reconfigurable routing resources a producer and a consumer of data that operate concurrently."



(12) **United States Patent**
Hammes

(10) **Patent No.:** **US 8,589,666 B2**
(45) **Date of Patent:** **Nov. 19, 2013**

(54) **ELIMINATION OF STREAM CONSUMER LOOP OVERSHOOT EFFECTS**

(75) Inventor: **Jeffrey Hammes**, Colorado Springs, CO (US)

(73) Assignee: **SRC Computers, Inc.**, Colorado Springs, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1985 days.

(21) Appl. No.: **11/456,466**

(22) Filed: **Jul. 10, 2006**

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,085,955 B2 *	8/2006	Prabhu	....	714/6
2002/0124159 A1 *	9/2002	Bekooji et al.	....	712/226

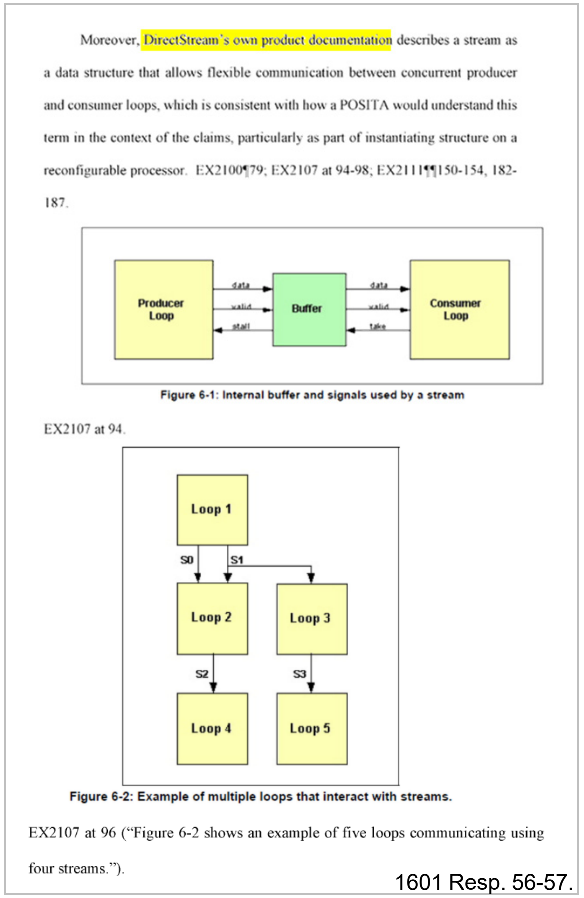
* cited by examiner

Primary Examiner — Andrew Caldwell
Assistant Examiner — George Giroux
(74) *Attorney, Agent, or Firm* — William J. Kubida; Hogan Lovells US LLP

(57) **ABSTRACT**

A reconfigurable processor invoking data stream pipelining is configured to associate a restore buffer with each incoming data stream. The buffer is configured to be of sufficient size to

Ex. 2027, Face.



Moreover, DirectStream's own product documentation describes a stream as a data structure that allows flexible communication between concurrent producer and consumer loops, which is consistent with how a POSITA would understand this term in the context of the claims, particularly as part of instantiating structure on a reconfigurable processor. EX2100¶79; EX2107 at 94-98; EX2111¶¶150-154, 182-187.

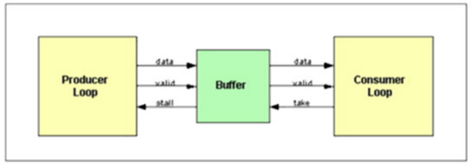


Figure 6-1: Internal buffer and signals used by a stream

EX2107 at 94.

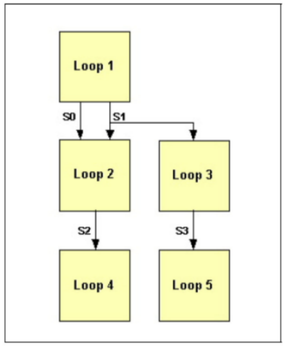


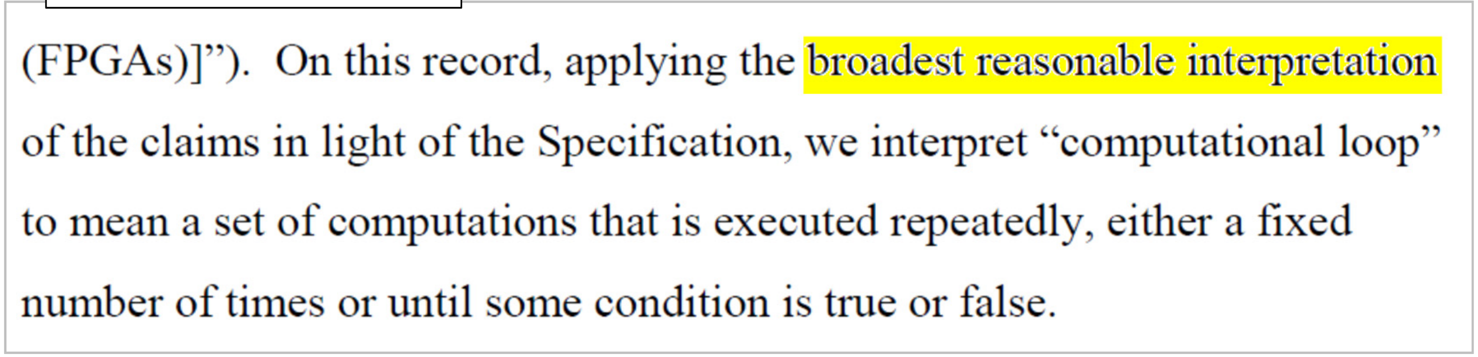
Figure 6-2: Example of multiple loops that interact with streams.

EX2107 at 96 ("Figure 6-2 shows an example of five loops communicating using four streams.").

1601 Resp. 56-57.

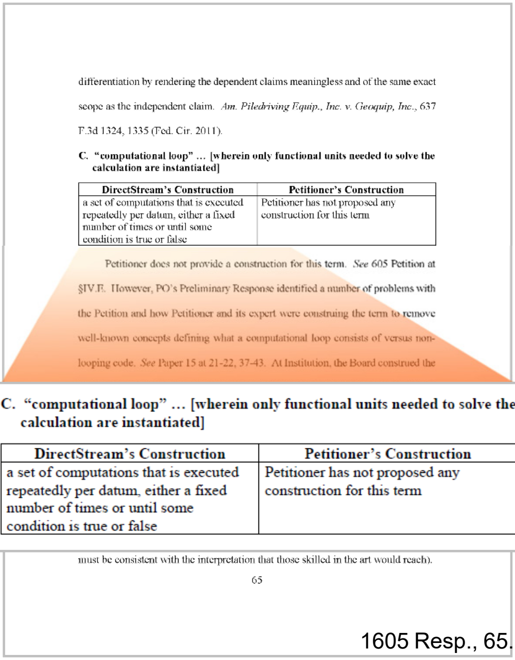
Claim Construction – Computational Loop

Board's Interpretation



(FPGAs)]"). On this record, applying the broadest reasonable interpretation of the claims in light of the Specification, we interpret "computational loop" to mean a set of computations that is executed repeatedly, either a fixed number of times or until some condition is true or false.

1601 Institution Decision, 23.



differentiation by rendering the dependent claims meaningless and of the same exact scope as the independent claim. *Am. Piledriving Equip., Inc. v. Geoquip, Inc.*, 637 F.3d 1324, 1335 (Fed. Cir. 2011).

C. "computational loop" … [wherein only functional units needed to solve the calculation are instantiated]

DirectStream's Construction	Petitioner's Construction
a set of computations that is executed repeatedly per datum, either a fixed number of times or until some condition is true or false	Petitioner has not proposed any construction for this term

Petitioner does not provide a construction for this term. *See* 605 Petition at §IV.E. However, PO's Preliminary Response identified a number of problems with the Petition and how Petitioner and its expert were construing the term to remove well-known concepts defining what a computational loop consists of versus non-looping code. *See* Paper 15 at 21-22, 37-43. At Institution, the Board construed the

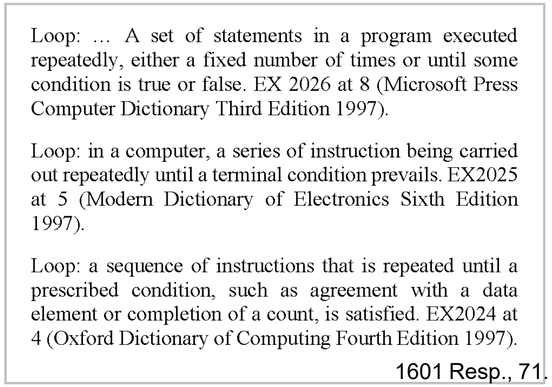
C. "computational loop" … [wherein only functional units needed to solve the calculation are instantiated]

DirectStream's Construction	Petitioner's Construction
a set of computations that is executed repeatedly per datum, either a fixed number of times or until some condition is true or false	Petitioner has not proposed any construction for this term

must be consistent with the interpretation that those skilled in the art would reach).

65

1605 Resp., 65.

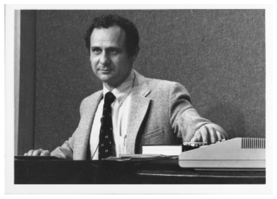


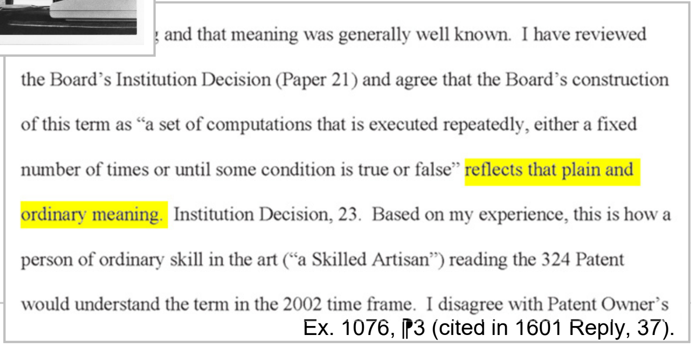
Loop: … A set of statements in a program executed repeatedly, either a fixed number of times or until some condition is true or false. EX 2026 at 8 (Microsoft Press Computer Dictionary Third Edition 1997).

Loop: in a computer, a series of instruction being carried out repeatedly until a terminal condition prevails. EX2025 at 5 (Modern Dictionary of Electronics Sixth Edition 1997).

Loop: a sequence of instructions that is repeated until a prescribed condition, such as agreement with a data element or completion of a count, is satisfied. EX2024 at 4 (Oxford Dictionary of Computing Fourth Edition 1997).

1601 Resp., 71.

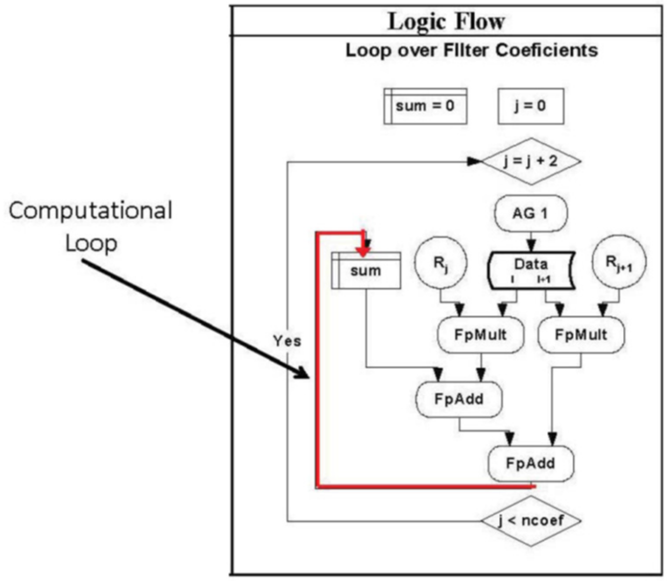




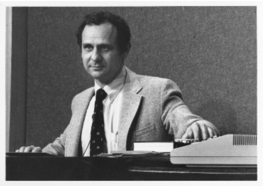
and that meaning was generally well known. I have reviewed the Board's Institution Decision (Paper 21) and agree that the Board's construction of this term as "a set of computations that is executed repeatedly, either a fixed number of times or until some condition is true or false" reflects that plain and ordinary meaning. Institution Decision, 23. Based on my experience, this is how a person of ordinary skill in the art ("a Skilled Artisan") reading the 324 Patent would understand the term in the 2002 time frame. I disagree with Patent Owner's

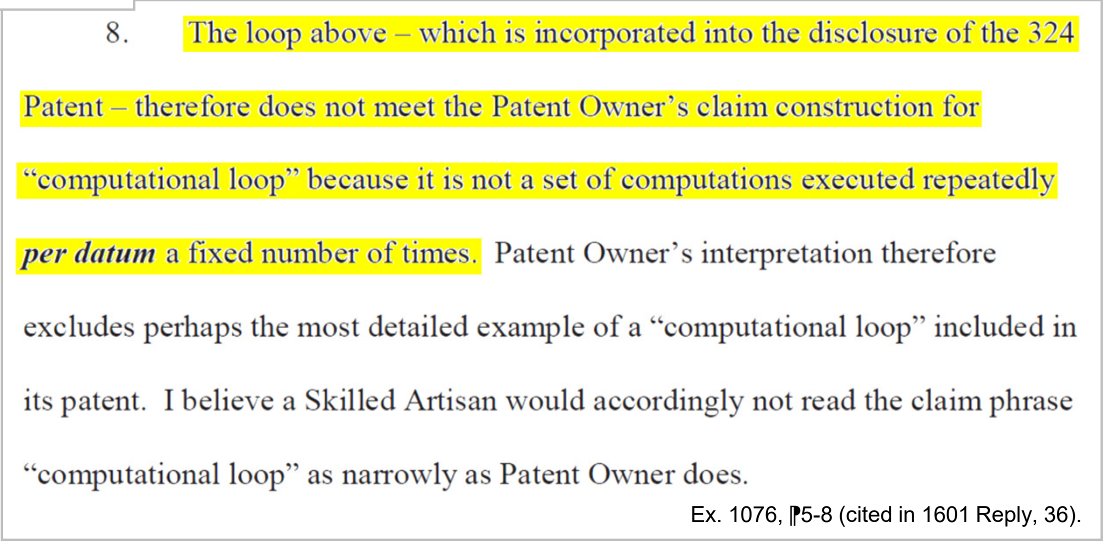
Ex. 1076, ⁋3 (cited in 1601 Reply, 37).

Claim Construction – Computational Loop



Ex. 1076, ⁋5 (cited in 1601 Reply, 36).

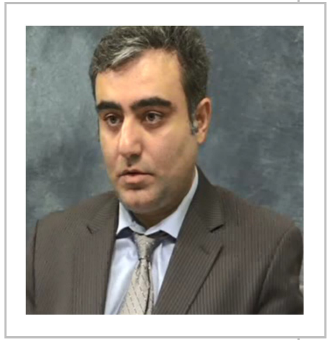


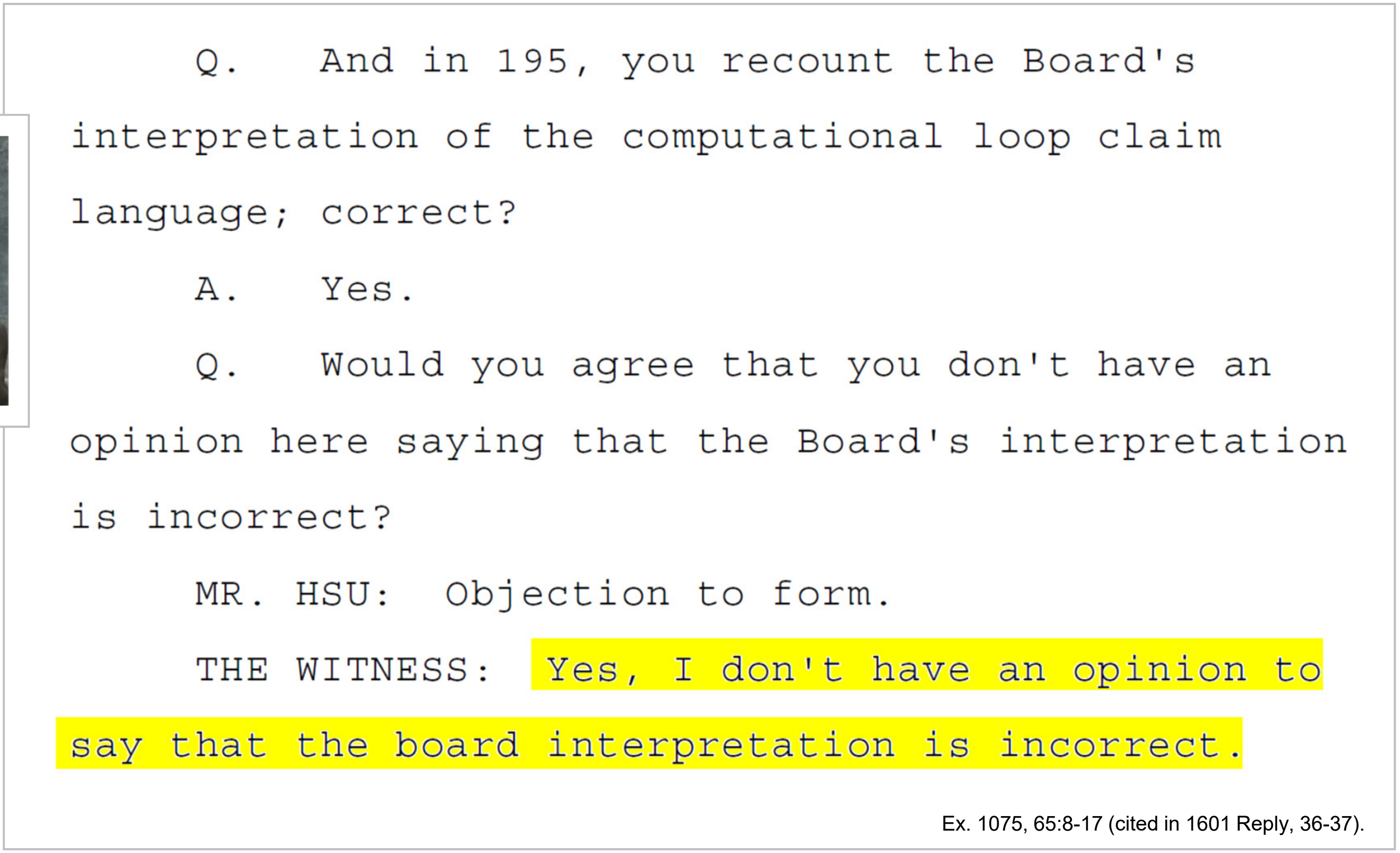


8. The loop above – which is incorporated into the disclosure of the 324 Patent – therefore does not meet the Patent Owner's claim construction for "computational loop" because it is not a set of computations executed repeatedly ***per datum*** a fixed number of times. Patent Owner's interpretation therefore excludes perhaps the most detailed example of a "computational loop" included in its patent. I believe a Skilled Artisan would accordingly not read the claim phrase "computational loop" as narrowly as Patent Owner does.

Ex. 1076, ⁋5-8 (cited in 1601 Reply, 36).

Claim Construction – Computational Loop





Q. And in 195, you recount the Board's interpretation of the computational loop claim language; correct?

A. Yes.

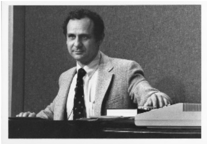
Q. Would you agree that you don't have an opinion here saying that the Board's interpretation is incorrect?

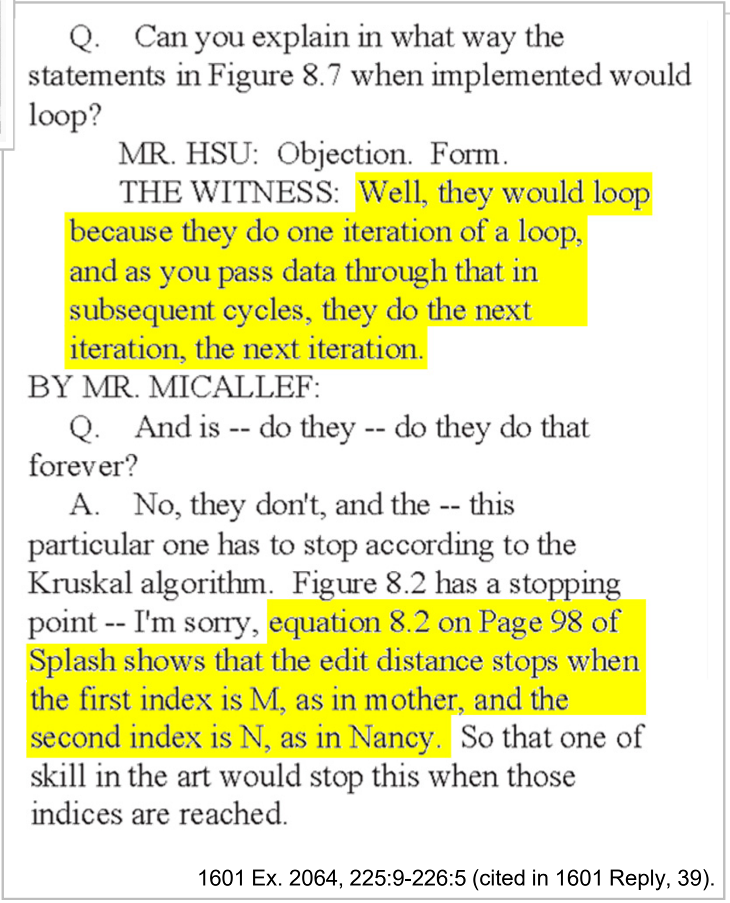
MR. HSU: Objection to form.

THE WITNESS: Yes, I don't have an opinion to say that the board interpretation is incorrect.

Ex. 1075, 65:8-17 (cited in 1601 Reply, 36-37).

Splash2 Computational Loops





Q. Can you explain in what way the statements in Figure 8.7 when implemented would loop?

MR. HSU: Objection. Form.

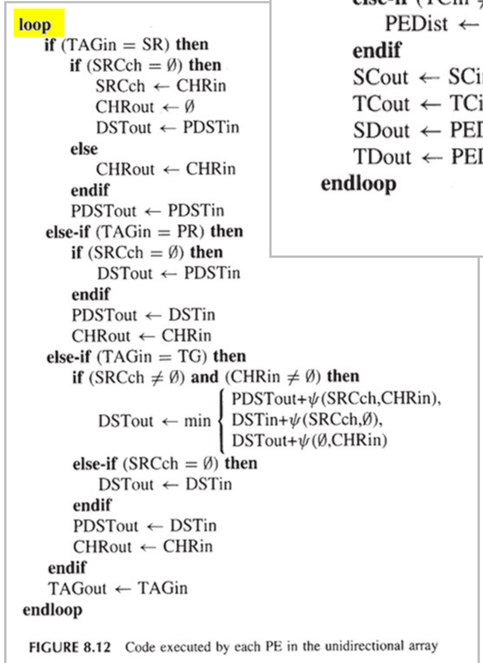
THE WITNESS: Well, they would loop because they do one iteration of a loop, and as you pass data through that in subsequent cycles, they do the next iteration, the next iteration.

BY MR. MICALLEF:

Q. And is -- do they -- do they do that forever?

A. No, they don't, and the -- this particular one has to stop according to the Kruskal algorithm. Figure 8.2 has a stopping point -- I'm sorry, equation 8.2 on Page 98 of Splash shows that the edit distance stops when the first index is M, as in mother, and the second index is N, as in Nancy. So that one of skill in the art would stop this when those indices are reached.

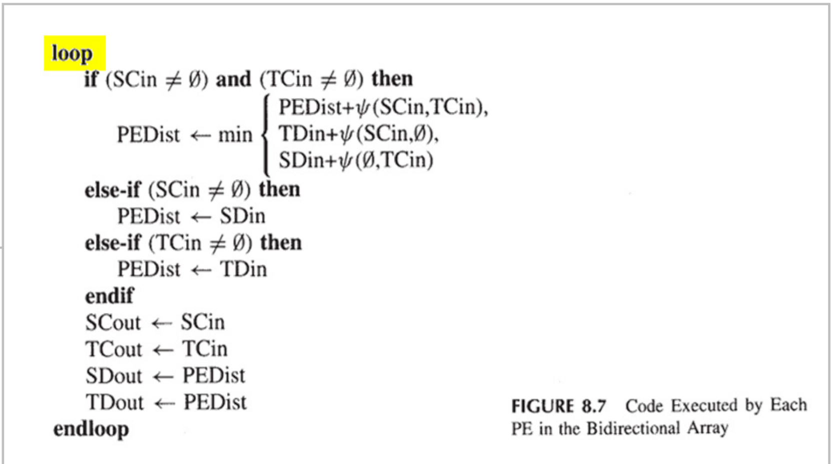
1601 Ex. 2064, 225:9-226:5 (cited in 1601 Reply, 39).



```
loop
  if (TAGin = SR) then
    if (SRCch = $\emptyset$) then
      SRCch ← CHRin
      CHRout ← $\emptyset$
      DSTout ← PDSTin
    else
      CHRout ← CHRin
    endif
    PDSTout ← PDSTin
  else-if (TAGin = PR) then
    if (SRCch = $\emptyset$) then
      DSTout ← PDSTin
    endif
    PDSTout ← DSTin
    CHRout ← CHRin
  else-if (TAGin = TG) then
    if (SRCch $\neq \emptyset$) and (CHRin $\neq \emptyset$) then
      DSTout ← min $\begin{cases} \text{PDSTout}+\psi(\text{SRCch},\text{CHRin}), \\ \text{DSTin}+\psi(\text{SRCch},\emptyset), \\ \text{DSTout}+\psi(\emptyset,\text{CHRin}) \end{cases}$
    else-if (SRCch = $\emptyset$) then
      DSTout ← DSTin
    endif
    PDSTout ← DSTin
    CHRout ← CHRin
  endif
  TAGout ← TAGin
endloop
```

FIGURE 8.12 Code executed by each PE in the unidirectional array

Ex. 1007, Fig. 8.12 (cited in 1601 Reply, 6).

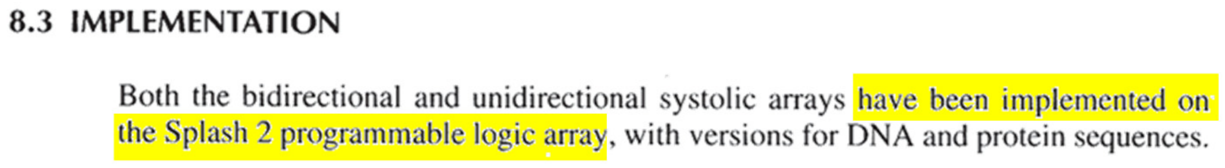


```
loop
  if (SCin $\neq \emptyset$) and (TCin $\neq \emptyset$) then
    PEDist ← min $\begin{cases} \text{PEDist}+\psi(\text{SCin},\text{TCin}), \\ \text{TDin}+\psi(\text{SCin},\emptyset), \\ \text{SDin}+\psi(\emptyset,\text{TCin}) \end{cases}$
  else-if (SCin $\neq \emptyset$) then
    PEDist ← SDin
  else-if (TCin $\neq \emptyset$) then
    PEDist ← TDin
  endif
  SCout ← SCin
  TCout ← TCin
  SDout ← PEDist
  TDout ← PEDist
endloop
```

FIGURE 8.7 Code Executed by Each PE in the Bidirectional Array

Ex. 1007, Fig. 8.7 (cited in 1601 Pet., 26).

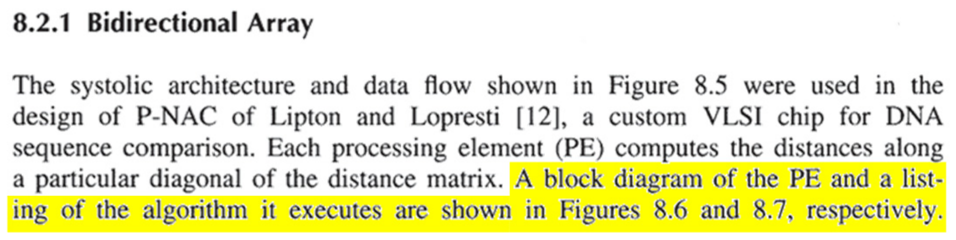
Splash2: Looping in the FPGAs



8.3 IMPLEMENTATION

Both the bidirectional and unidirectional systolic arrays have been implemented on the Splash 2 programmable logic array, with versions for DNA and protein sequences.

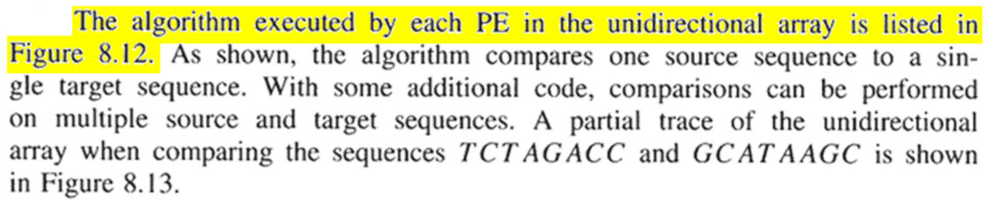
Ex. 1007, 104 (cited in 1601 Reply, 17).



8.2.1 Bidirectional Array

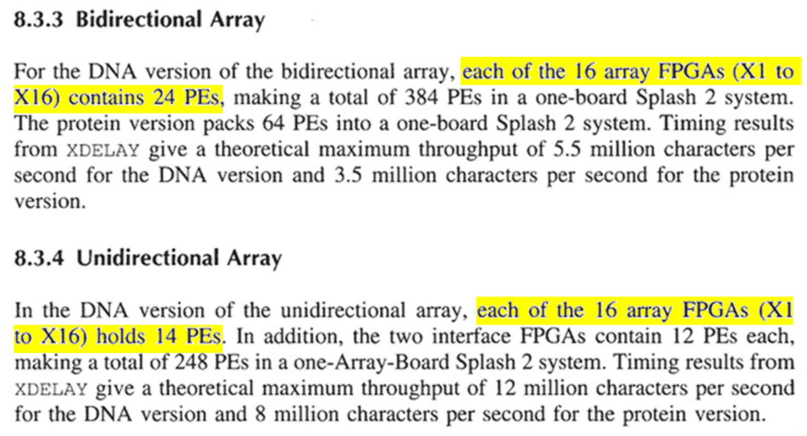
The systolic architecture and data flow shown in Figure 8.5 were used in the design of P-NAC of Lipton and Lopresti [12], a custom VLSI chip for DNA sequence comparison. Each processing element (PE) computes the distances along a particular diagonal of the distance matrix. A block diagram of the PE and a listing of the algorithm it executes are shown in Figures 8.6 and 8.7, respectively.

Ex. 1007, 100 (cited in 1601 Reply, 17-18).



The algorithm executed by each PE in the unidirectional array is listed in Figure 8.12. As shown, the algorithm compares one source sequence to a single target sequence. With some additional code, comparisons can be performed on multiple source and target sequences. A partial trace of the unidirectional array when comparing the sequences *TCTAGACC* and *GCATAAGC* is shown in Figure 8.13.

Ex. 1007, 104 (cited in 1601 Reply, 18).



8.3.3 Bidirectional Array

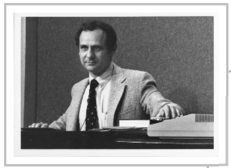
For the DNA version of the bidirectional array, each of the 16 array FPGAs (X1 to X16) contains 24 PEs, making a total of 384 PEs in a one-board Splash 2 system. The protein version packs 64 PEs into a one-board Splash 2 system. Timing results from `XDELAY` give a theoretical maximum throughput of 5.5 million characters per second for the DNA version and 3.5 million characters per second for the protein version.

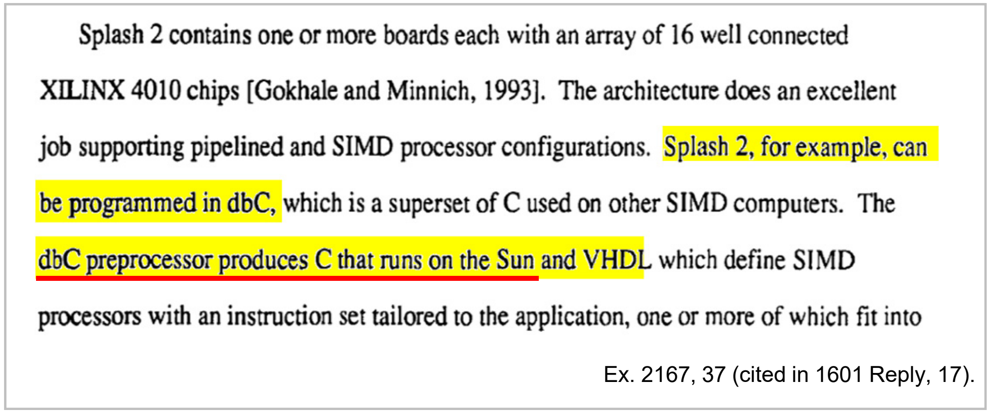
8.3.4 Unidirectional Array

In the DNA version of the unidirectional array, each of the 16 array FPGAs (X1 to X16) holds 14 PEs. In addition, the two interface FPGAs contain 12 PEs each, making a total of 248 PEs in a one-Array-Board Splash 2 system. Timing results from `XDELAY` give a theoretical maximum throughput of 12 million characters per second for the DNA version and 8 million characters per second for the protein version.

Ex. 1007, 107 (cited in 1601 Reply, 18).

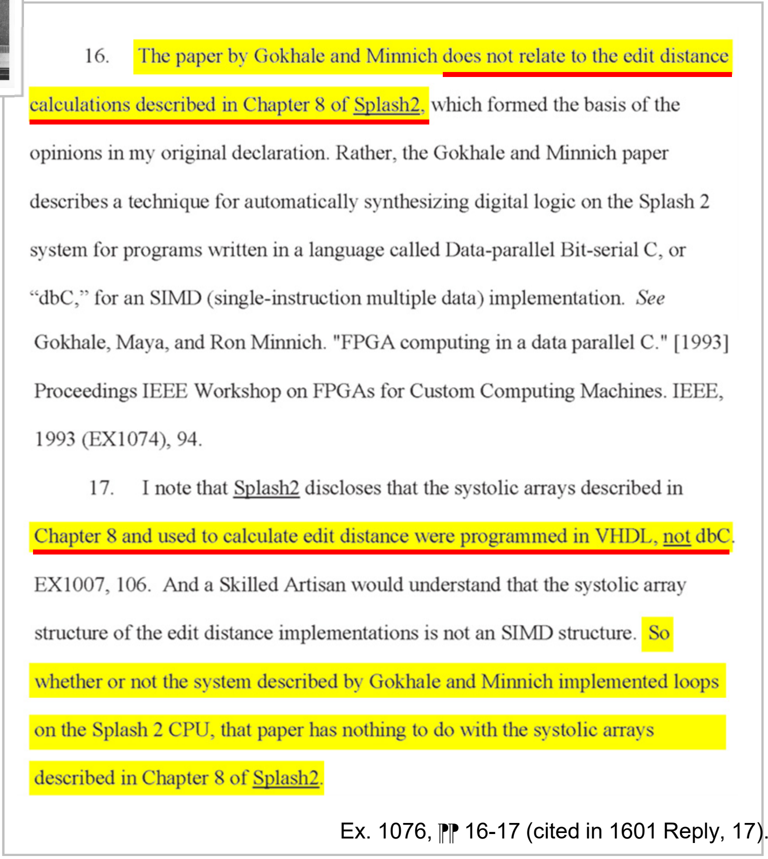
Splash2: Looping in the FPGAs





Splash 2 contains one or more boards each with an array of 16 well connected XILINX 4010 chips [Gokhale and Minnich, 1993]. The architecture does an excellent job supporting pipelined and SIMD processor configurations. Splash 2, for example, can be programmed in dbC, which is a superset of C used on other SIMD computers. The dbC preprocessor produces C that runs on the Sun and VHDL which define SIMD processors with an instruction set tailored to the application, one or more of which fit into

Ex. 2167, 37 (cited in 1601 Reply, 17).



16. The paper by Gokhale and Minnich does not relate to the edit distance calculations described in Chapter 8 of Splash2, which formed the basis of the opinions in my original declaration. Rather, the Gokhale and Minnich paper describes a technique for automatically synthesizing digital logic on the Splash 2 system for programs written in a language called Data-parallel Bit-serial C, or "dbC," for an SIMD (single-instruction multiple data) implementation. *See* Gokhale, Maya, and Ron Minnich. "FPGA computing in a data parallel C." [1993] Proceedings IEEE Workshop on FPGAs for Custom Computing Machines. IEEE, 1993 (EX1074), 94.

17. I note that Splash2 discloses that the systolic arrays described in Chapter 8 and used to calculate edit distance were programmed in VHDL, not dbC. EX1007, 106. And a Skilled Artisan would understand that the systolic array structure of the edit distance implementations is not an SIMD structure. So whether or not the system described by Gokhale and Minnich implemented loops on the Splash 2 CPU, that paper has nothing to do with the systolic arrays described in Chapter 8 of Splash2.

Ex. 1076, ¶¶ 16-17 (cited in 1601 Reply, 17).

RaPiD Computational Loops

$$z_{mj} = \sum_{n=0}^{N-1} a_{mn} w_{nj}, \qquad (5)$$

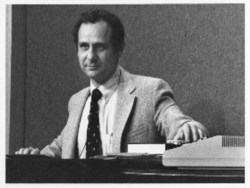
and thus

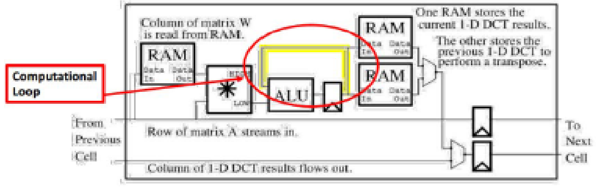
$$y_{ji} = \sum_{m=0}^{N-1} z_{mj} w_{mi} \qquad (6)$$

Ex, 1009, 111 (cited in 1601 Reply, 41-42).

Thus, the equations set forth above calculate a running sum, with the intermediate sum output by each iteration fed back as an input to the next iteration of calculations.

Ex. 1076, ⁋22 (cited in 1601 Reply, 42).





Computational Loop
Column of matrix W is read from RAM.
RAM
Data In
Data Out
HIGH
LOW
ALU
RAM
RAM
One RAM stores the current 1-D DCT results.
The other stores the previous 1-D DCT to perform a transpose.
From Previous Cell
Row of matrix A streams in.
Column of 1-D DCT results flows out.
To Next Cell

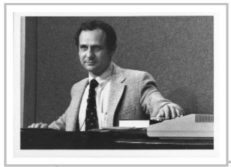
Figure 10: *Netlist for one cell of 2-D DCT. The top pipelined bus streams in the* **A** *matrix while the bottom bus streams out resulting 1-D DCT, transposed. The top bus also streams the* **W** *columns into the local memories prior to the computation.*

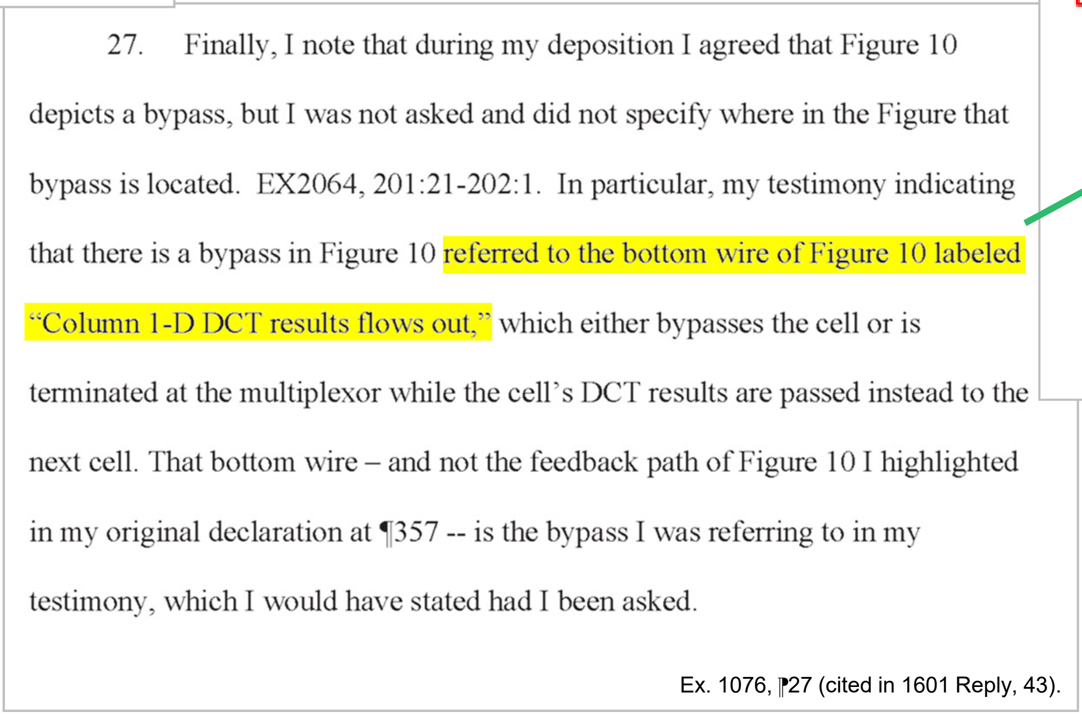
Ex, 1009, Fig. 10 (cited in 1601 Reply, 43).

24. To illustrate how the calculations set forth above are carried out repeatedly, I highlighted in an annotated version of Figure 10 in ¶357 of my original declaration the location in the hardware where the output of the ALU (*i.e.*, the running sum) is looped back to the ALU input for use in the next iteration of the loop:

Ex. 1076, ⁋24 (cited in 1601, Reply 42-43).

RaPiD Computational Loops





27. Finally, I note that during my deposition I agreed that Figure 10 depicts a bypass, but I was not asked and did not specify where in the Figure that bypass is located. EX2064, 201:21-202:1. In particular, my testimony indicating that there is a bypass in Figure 10 referred to the bottom wire of Figure 10 labeled "Column 1-D DCT results flows out," which either bypasses the cell or is terminated at the multiplexor while the cell's DCT results are passed instead to the next cell. That bottom wire – and not the feedback path of Figure 10 I highlighted in my original declaration at ¶357 -- is the bypass I was referring to in my testimony, which I would have stated had I been asked.

Ex. 1076, ⁋27 (cited in 1601 Reply, 43).

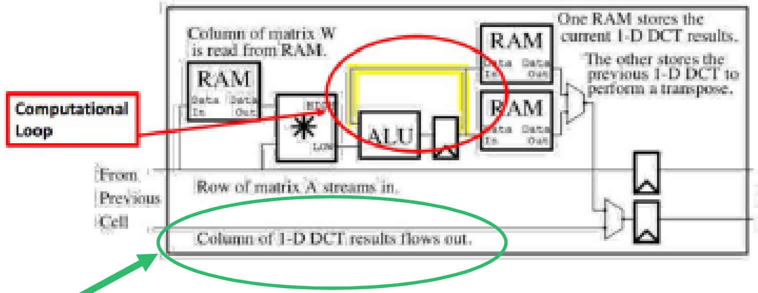
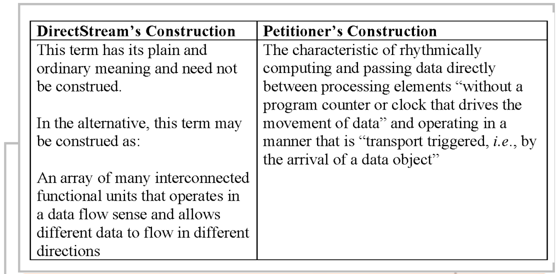

Computational Loop
Column of matrix W is read from RAM.
RAM
Data In
Data Out
One RAM stores the current 1-D DCT results.
The other stores the previous 1-D DCT to perform a transpose.
ALU
From Previous Cell
Row of matrix A streams in.
Column of 1-D DCT results flows out.
To Next Cell

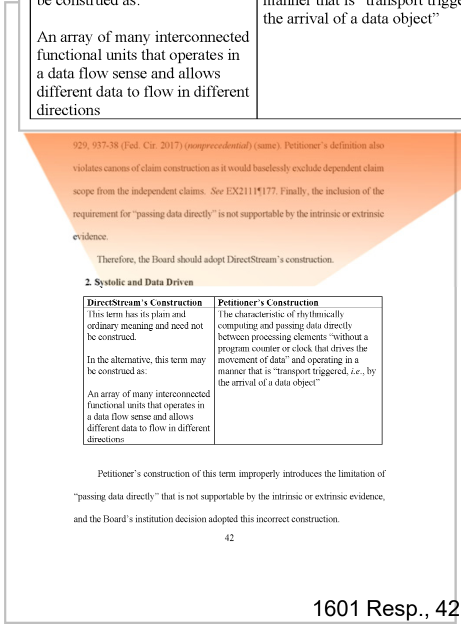
Figure 10: *Netlist for one cell of 2-D DCT. The top pipelined bus streams in the* **A** *matrix while the bottom bus streams out resulting 1-D DCT, transposed. The top bus also streams the* **W** *columns into the local memories prior to the computation.*

Ex, 1009, Fig. 10 (cite in 1601 Reply, 43).

Claim Construction - Systolic



DirectStream's Construction	Petitioner's Construction
This term has its plain and ordinary meaning and need not be construed. In the alternative, this term may be construed as: An array of many interconnected functional units that operates in a data flow sense and allows different data to flow in different directions	The characteristic of rhythmically computing and passing data directly between processing elements "without a program counter or clock that drives the movement of data" and operating in a manner that is "transport triggered, *i.e.*, by the arrival of a data object"



929, 937-38 (Fed. Cir. 2017) (*nonprecedential*) (same). Petitioner's definition also violates canons of claim construction as it would baselessly exclude dependent claim scope from the independent claims. *See* EX2111¶177. Finally, the inclusion of the requirement for "passing data directly" is not supportable by the intrinsic or extrinsic evidence.

Therefore, the Board should adopt DirectStream's construction.

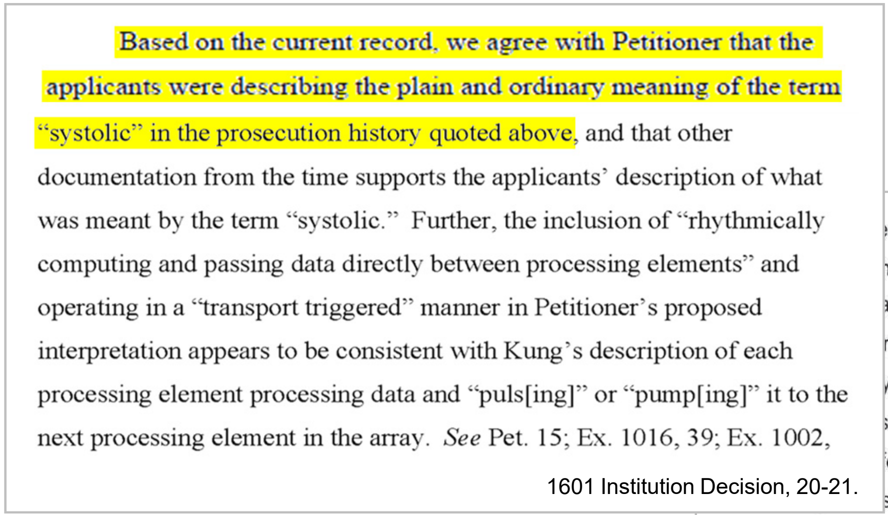
2. Systolic and Data Driven

DirectStream's Construction	Petitioner's Construction
This term has its plain and ordinary meaning and need not be construed. In the alternative, this term may be construed as: An array of many interconnected functional units that operates in a data flow sense and allows different data to flow in different directions	The characteristic of rhythmically computing and passing data directly between processing elements "without a program counter or clock that drives the movement of data" and operating in a manner that is "transport triggered, *i.e.*, by the arrival of a data object"

Petitioner's construction of this term improperly introduces the limitation of "passing data directly" that is not supportable by the intrinsic or extrinsic evidence, and the Board's institution decision adopted this incorrect construction.

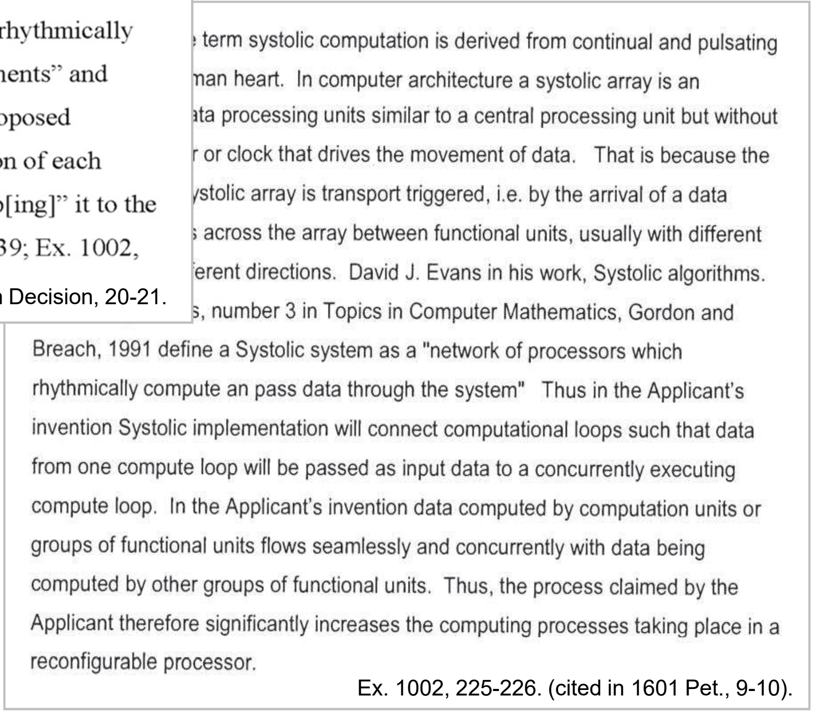
42

1601 Resp., 42.



Based on the current record, we agree with Petitioner that the applicants were describing the plain and ordinary meaning of the term "systolic" in the prosecution history quoted above, and that other documentation from the time supports the applicants' description of what was meant by the term "systolic." Further, the inclusion of "rhythmically computing and passing data directly between processing elements" and operating in a "transport triggered" manner in Petitioner's proposed interpretation appears to be consistent with Kung's description of each processing element processing data and "puls[ing]" or "pump[ing]" it to the next processing element in the array. *See* Pet. 15; Ex. 1016, 39; Ex. 1002,

1601 Institution Decision, 20-21.

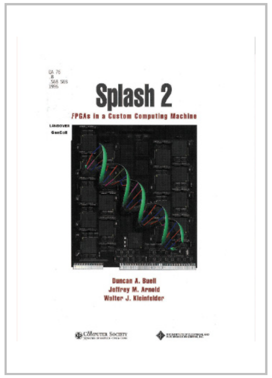


term systolic computation is derived from continual and pulsating
man heart. In computer architecture a systolic array is an
ata processing units similar to a central processing unit but without
r or clock that drives the movement of data. That is because the
ystolic array is transport triggered, i.e. by the arrival of a data
across the array between functional units, usually with different
erent directions. David J. Evans in his work, Systolic algorithms.
s, number 3 in Topics in Computer Mathematics, Gordon and
Breach, 1991 define a Systolic system as a "network of processors which rhythmically compute an pass data through the system" Thus in the Applicant's invention Systolic implementation will connect computational loops such that data from one compute loop will be passed as input data to a concurrently executing compute loop. In the Applicant's invention data computed by computation units or groups of functional units flows seamlessly and concurrently with data being computed by other groups of functional units. Thus, the process claimed by the Applicant therefore significantly increases the computing processes taking place in a reconfigurable processor.

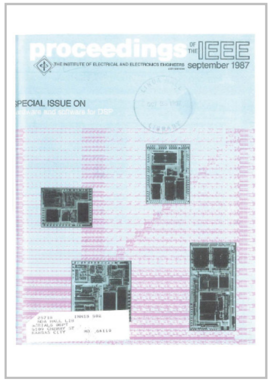
Ex. 1002, 225-226. (cited in 1601 Pet., 9-10).

Petitioner's Demonstrative – Not Evidence

The Obvious Combination – Splash2/Gaudiot



Ex. 1007, Face

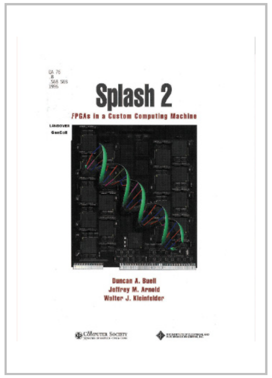


Ex. 1010 (Gaudiot), Face

- Unrebutted Reasons to Combine
  - Analogous art
  - Arrangement of old elements; predictable results
  - Gaudiot's techniques offer **increased flexibility** due to scheduling
  - Gaudiot's techniques "possess[ed] no notion of central control and can deliver **maximum parallelism in very complex algorithms**"
  - Data driven techniques in the systolic arrays of Splash2 "present[ed] **the crucial advantage of scalability**"
  - The programmability afforded by this approach translates into a **higher performance** for a given amount of programming effort

1601 Ex. 1003 ⁋⁋200-202 (cited in 1601 Pet., 54-55); *see also* Reply, 9, 51.

The Obvious Combination – Splash2/RaPiD

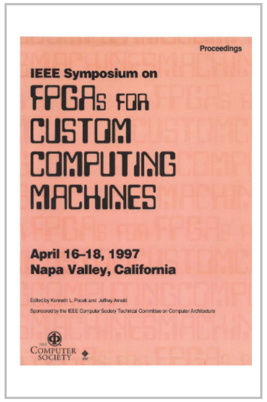


Splash 2

FPGAs in a Custom Computing Machine

Duncan A. Buell
Jeffrey M. Arnold
Walter J. Kleinfelder

Ex. 1007, Face



Proceedings

IEEE Symposium on

FPGAs for Custom Computing Machines

April 16–18, 1997
Napa Valley, California

Edited by Kenneth L. Pocek and Jeffrey Arnold

Sponsored by the IEEE Computer Society Technical Committee on Computer Architecture

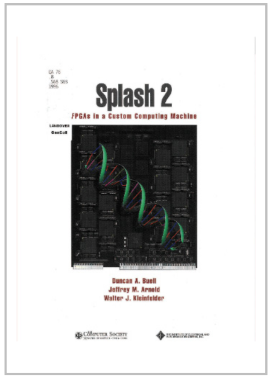
Computer Society

Ex. 1009 (RaPiD), Face

- Unrebutted Reasons to Combine
  - Analogous art
  - Arrangement of old elements; predictable results
  - <u>RaPiD</u> cites <u>Splash2</u> as a "**very successful** example[] of a reconfigurable system"
  - **Increasing popularity** of image compression techniques that employed the DCT
  - Splash 2 platform "possesses architectural properties that make it **well suited for the computation and data transfer rates** that are characteristic of this class of problems. Furthermore, the price/performance of this system makes it a competitive alternative."
  - Splash 2 platform has **advantages for image processing techniques**.
  - Splash 2 platform provides "**a flexible interface** design that facilitates customized I/O."

1601 Ex. 1003 ¶¶364-369 (cited in 1601 Pet., 65-67); *see also* Reply, 9.

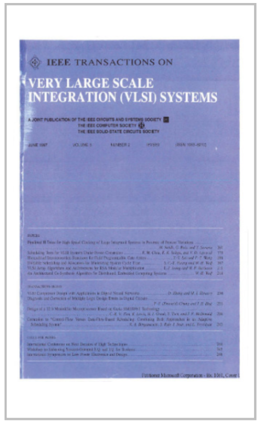
The Obvious Combination – Splash2/Jeong


Splash 2

FPGAs in a Custom Computing Machine

Duncan A. Buell
Jeffrey M. Arnold
Walter J. Kleinfelder

Ex. 1007, Face

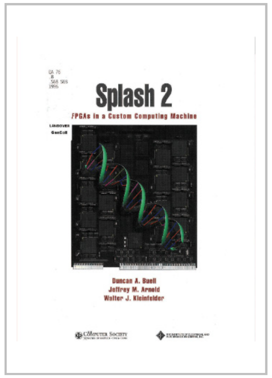

IEEE TRANSACTIONS ON
VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS

Ex. 1061 (Jeong), Face

- Unrebutted Reasons to Combine
  - Analogous art
  - Arrangement of old elements; predictable results
  - <u>Jeong</u> **maps his algorithms to systolic structures**
  - <u>Jeong</u> **discloses intent to use FPGAs** to implement.
  - **Increasing popularity** of systolic modular multiplication systems for data encryption

1601 Ex. 1003 ¶¶ 459-461 (cited in 1601 Pet., 76-77); *see also* Reply, 9.

The Obvious Combination – Splash2/Chunky SLD


Splash 2

FPGAs in a Custom Computing Machine

Duncan A. Buell
Jeffrey M. Arnold
Walter J. Kleinfelder

Ex. 1007, Face

Proceedings

IEEE Symposium on
FPGAs for
Custom
Computing
Machines

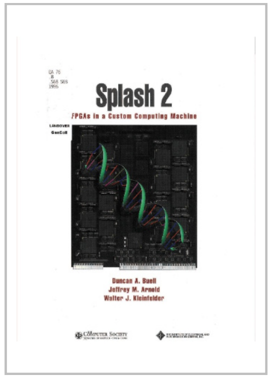
April 16–18, 1997
Napa Valley, California

Ex. 1011 (Chunky SLD), Face

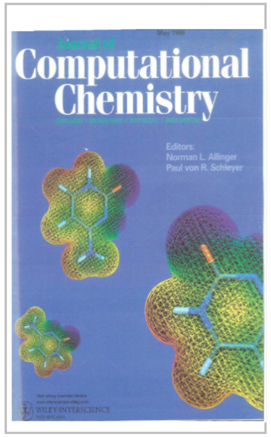
- Unrebutted Reasons to Combine
  - Analogous art
  - Arrangement of old elements; predictable results
  - Chunky SLD **expressly cites the Splash 2 system** as the platform to which its computing algorithms are mapped.
  - **Increasing popularity** of automatic target recognition systems.
  - Splash 2 platform "possesses architectural properties that make it **well suited for the computation and data transfer rates** that are characteristic of this class of problems. Furthermore, the price/performance of this system makes it a competitive alternative."
  - Splash 2 platform has characteristics that make it **advantageous for image processing** techniques, such as automated VHDL code.
  - Splash2 identifies **additional advantages of performing image processing on the Splash 2 platform**, such as "a flexible interface design that facilitates customized I/O," and noting that a particular image processing system has been constructed on Splash.

1602 Ex. 1003 ⁋⁋412-417 (cited in 1602 Pet., 74-76); *see also* Reply, 9.

The Obvious Combination – Splash2/Roccatano



Ex. 1007, Face

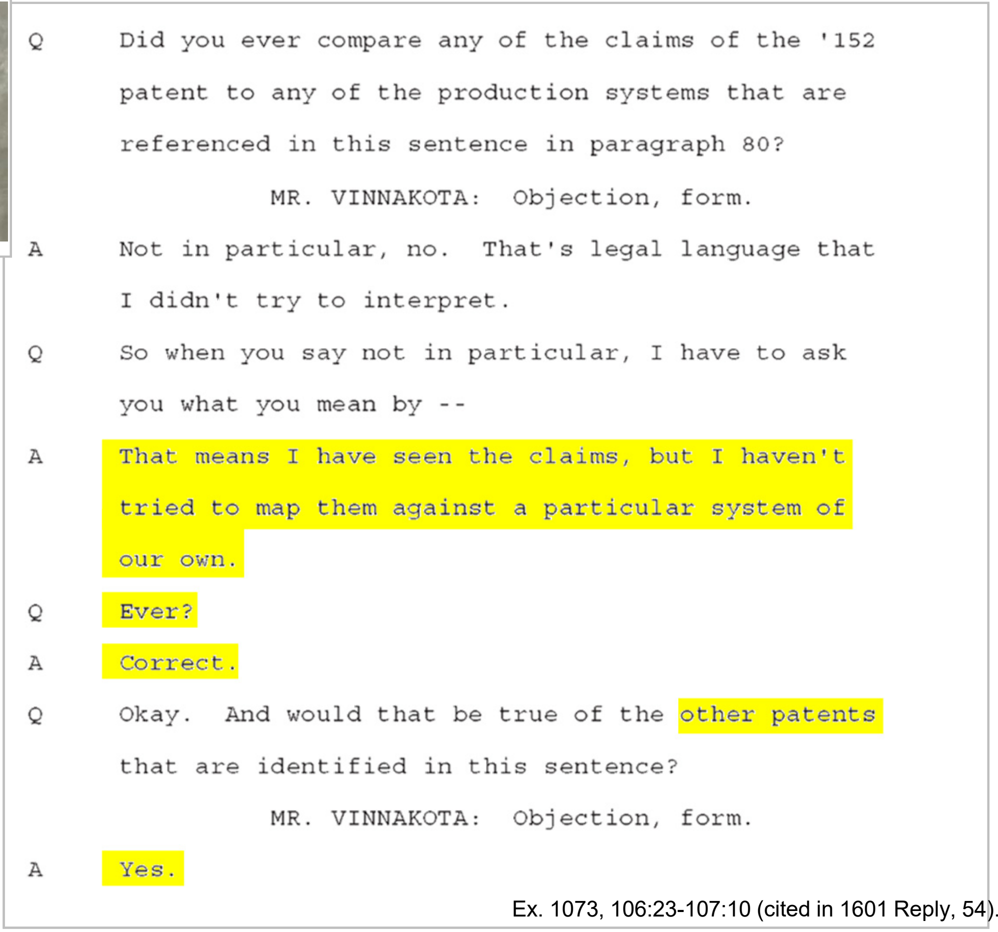


Ex. 1012 (Roccatano), Face

- Unrebutted Reasons to Combine
  - Analogous art
  - Arrangement of old elements; predictable results
  - Splash2 is one of two reconfigurable systems that have achieved "**supercomputer performance**" on applications that include molecular biology, which is the underlying application for Roccatano.
  - **Increasing popularity** of parallel computer simulation techniques for molecular dynamics

1603 Ex. 1003 ⁋⁋518-520 (cited in 1603 Pet., 77-79); *see also* Reply, 9.

No Secondary Considerations



Q Did you ever compare any of the claims of the '152 patent to any of the production systems that are referenced in this sentence in paragraph 80?

MR. VINNAKOTA: Objection, form.

A Not in particular, no. That's legal language that I didn't try to interpret.

Q So when you say not in particular, I have to ask you what you mean by --

A That means I have seen the claims, but I haven't tried to map them against a particular system of our own.

Q Ever?

A Correct.

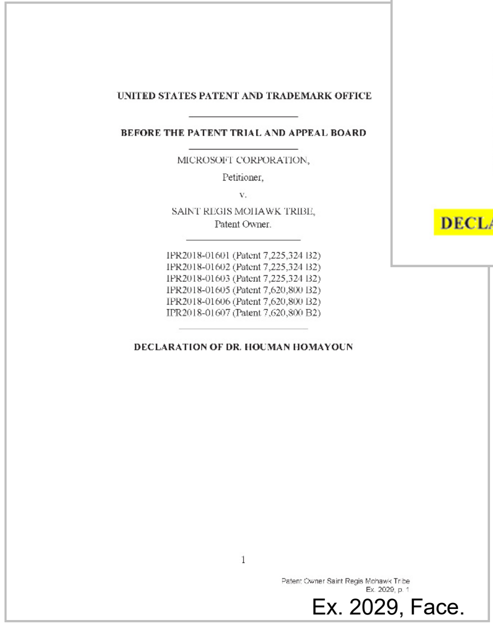
Q Okay. And would that be true of the other patents that are identified in this sentence?

MR. VINNAKOTA: Objection, form.

A Yes.

Ex. 1073, 106:23-107:10 (cited in 1601 Reply, 54).

The Proper Level of Skill



UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

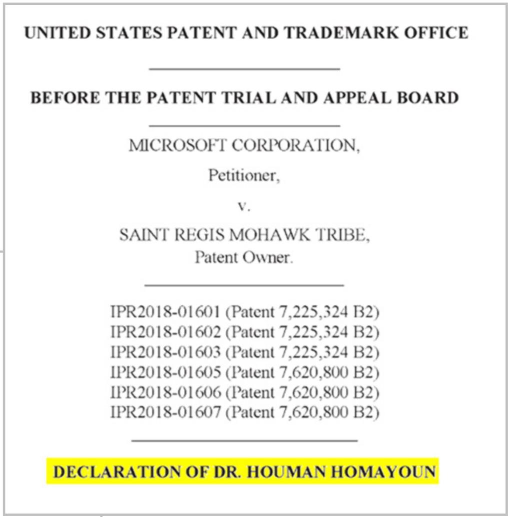
MICROSOFT CORPORATION,
Petitioner,
v.
SAINT REGIS MOHAWK TRIBE,
Patent Owner.

IPR2018-01601 (Patent 7,225,324 B2)
IPR2018-01602 (Patent 7,225,324 B2)
IPR2018-01603 (Patent 7,225,324 B2)
IPR2018-01605 (Patent 7,620,800 B2)
IPR2018-01606 (Patent 7,620,800 B2)
IPR2018-01607 (Patent 7,620,800 B2)

DECLARATION OF DR. HOUMAN HOMAYOUN

1

Patent Owner Saint Regis Mohawk Tribe
Ex. 2029, p. 1



UNITED STATES PATENT AND TRADEMARK OFFICE

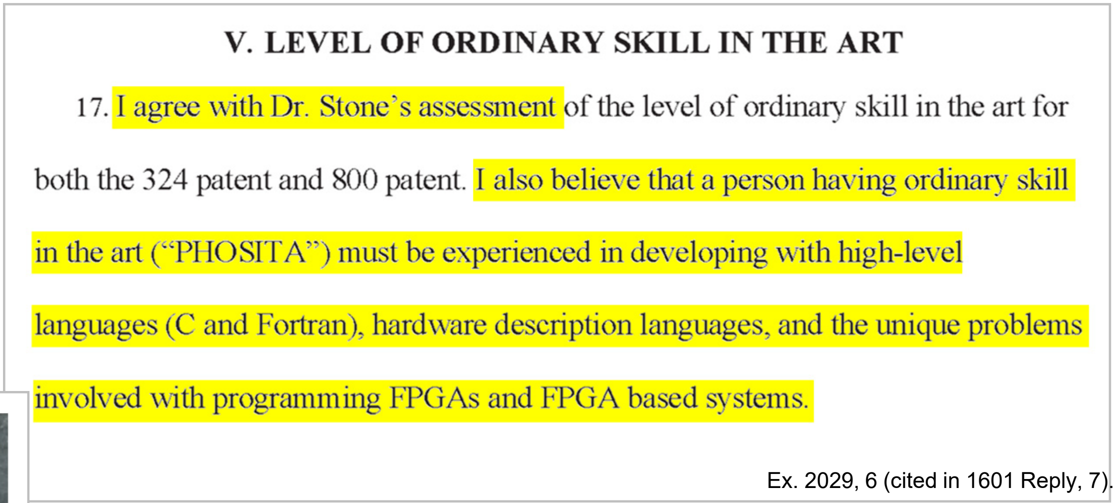
BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,
v.
SAINT REGIS MOHAWK TRIBE,
Patent Owner.

IPR2018-01601 (Patent 7,225,324 B2)
IPR2018-01602 (Patent 7,225,324 B2)
IPR2018-01603 (Patent 7,225,324 B2)
IPR2018-01605 (Patent 7,620,800 B2)
IPR2018-01606 (Patent 7,620,800 B2)
IPR2018-01607 (Patent 7,620,800 B2)

DECLARATION OF DR. HOUMAN HOMAYOUN

Ex. 2029, Face.

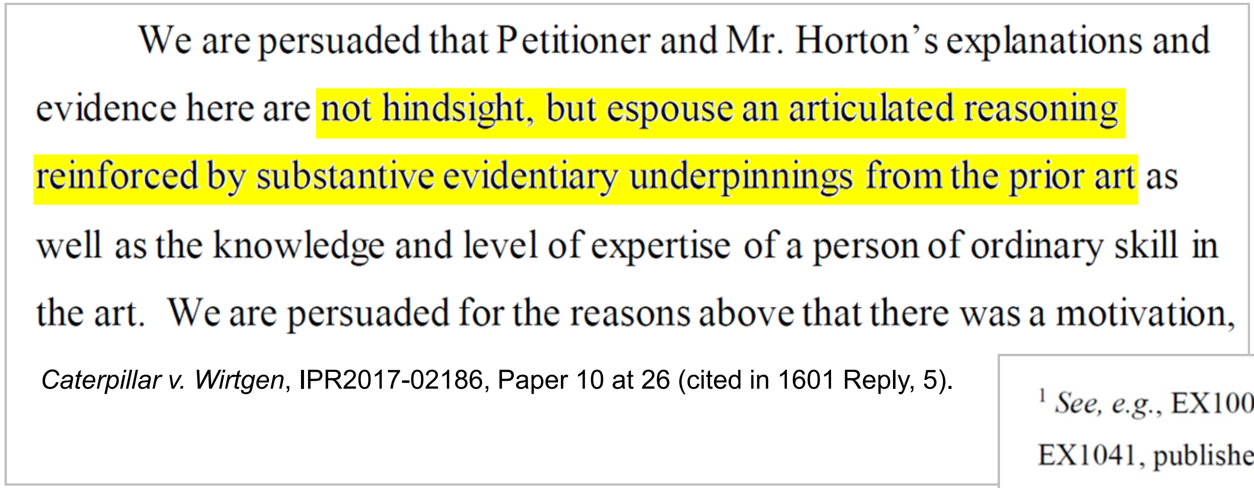


V. LEVEL OF ORDINARY SKILL IN THE ART

17. I agree with Dr. Stone's assessment of the level of ordinary skill in the art for both the 324 patent and 800 patent. I also believe that a person having ordinary skill in the art ("PHOSITA") must be experienced in developing with high-level languages (C and Fortran), hardware description languages, and the unique problems involved with programming FPGAs and FPGA based systems.

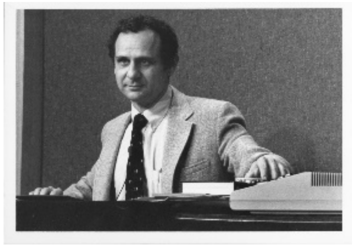
Ex. 2029, 6 (cited in 1601 Reply, 7).

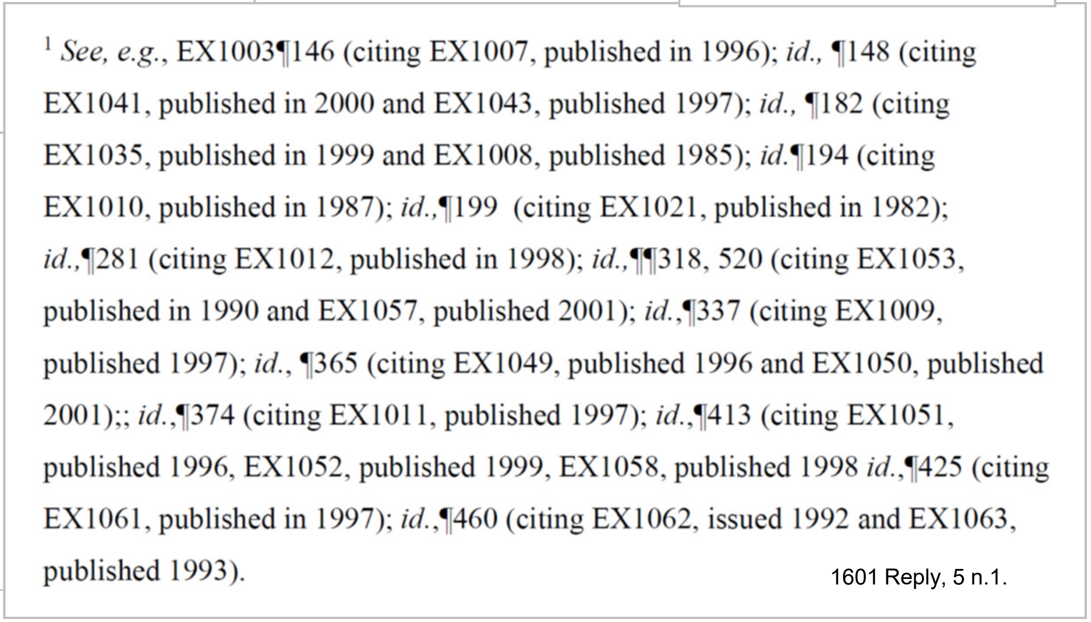
No Hindsight



We are persuaded that Petitioner and Mr. Horton's explanations and evidence here are not hindsight, but espouse an articulated reasoning reinforced by substantive evidentiary underpinnings from the prior art as well as the knowledge and level of expertise of a person of ordinary skill in the art. We are persuaded for the reasons above that there was a motivation,

Caterpillar v. Wirtgen, IPR2017-02186, Paper 10 at 26 (cited in 1601 Reply, 5).

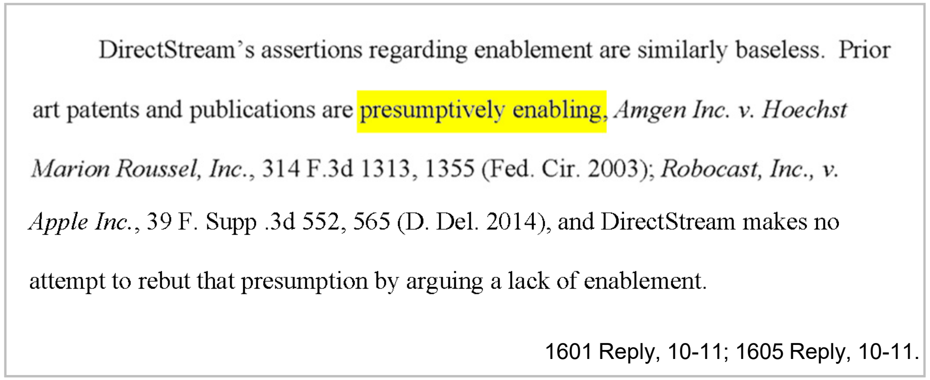




[1] *See, e.g.*, EX1003¶146 (citing EX1007, published in 1996); *id.*, ¶148 (citing EX1041, published in 2000 and EX1043, published 1997); *id.*, ¶182 (citing EX1035, published in 1999 and EX1008, published 1985); *id.*¶194 (citing EX1010, published in 1987); *id.*,¶199 (citing EX1021, published in 1982); *id.*,¶281 (citing EX1012, published in 1998); *id.*,¶¶318, 520 (citing EX1053, published in 1990 and EX1057, published 2001); *id.*,¶337 (citing EX1009, published 1997); *id.*, ¶365 (citing EX1049, published 1996 and EX1050, published 2001);; *id.*,¶374 (citing EX1011, published 1997); *id.*,¶413 (citing EX1051, published 1996, EX1052, published 1999, EX1058, published 1998 *id.*,¶425 (citing EX1061, published in 1997); *id.*,¶460 (citing EX1062, issued 1992 and EX1063, published 1993).

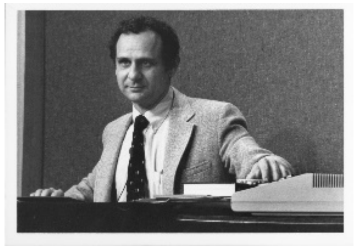
1601 Reply, 5 n.1.

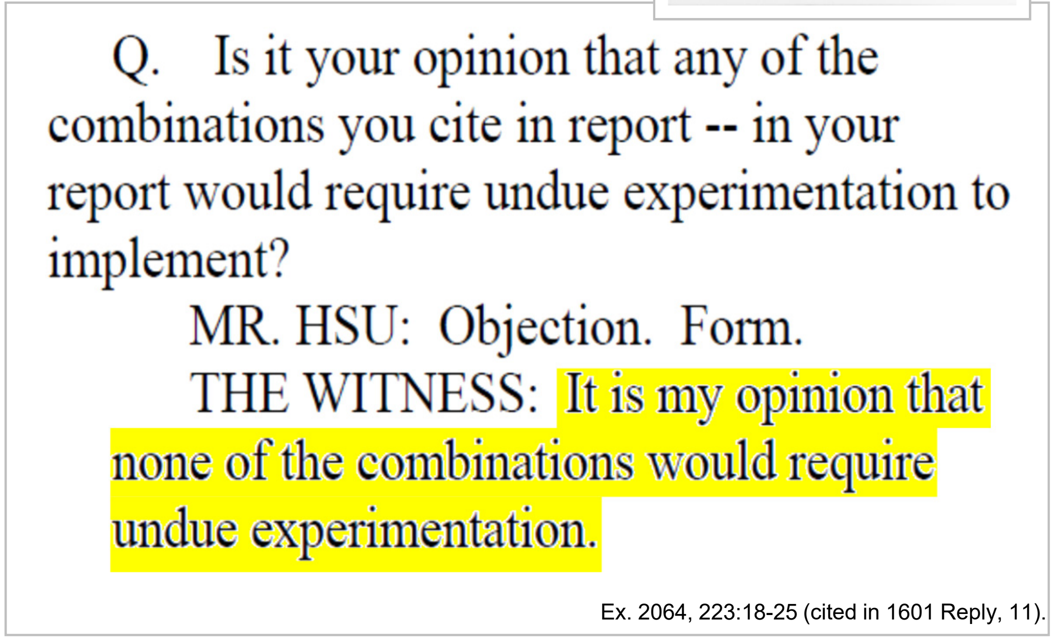
Enabled Prior Art



DirectStream's assertions regarding enablement are similarly baseless. Prior art patents and publications are presumptively enabling, *Amgen Inc. v. Hoechst Marion Roussel, Inc.*, 314 F.3d 1313, 1355 (Fed. Cir. 2003); *Robocast, Inc., v. Apple Inc.*, 39 F. Supp .3d 552, 565 (D. Del. 2014), and DirectStream makes no attempt to rebut that presumption by arguing a lack of enablement.

1601 Reply, 10-11; 1605 Reply, 10-11.





Q. Is it your opinion that any of the combinations you cite in report -- in your report would require undue experimentation to implement?

MR. HSU: Objection. Form.

THE WITNESS: It is my opinion that none of the combinations would require undue experimentation.

Ex. 2064, 223:18-25 (cited in 1601 Reply, 11).

Beijing

Boston

Brussels

Century City

Chicago

Dallas

Geneva

Hong Kong

Houston

London

Los Angeles

Munich

New York

Palo Alto

San Francisco

Shanghai

Singapore

Sydney

Tokyo

Washington, D.C.

sidley.com