

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,

v.

ATI TECHNOLOGIES ULC
Patent Owner.

Case No. IPR2023-00922
U.S. Patent No. 8,760,454

**PATENT OWNER'S RESPONSE
PURSUANT TO 37 C.F.R. § 42.107(A)**

Table of Contents

	Page
I. INTRODUCTION	1
II. GRAPHICS PROCESSING PRIMER	3
A. Graphics Processors Use Vertex and Pixel Data to Create Display Images .	3
B. Conventional Graphics Processors Use Separate Dedicated Vertex and Pixel Shaders	5
III. '454 PATENT OVERVIEW	7
A. The '454 Patent's Unified Shader Determines Which Data To Process By Evaluating Storage Capacity As Seen In Claims 1, 3-10	7
B. The Unified Shader Can Simultaneously Execute Vertex and Pixel Operations and Switch Quickly Between Operations at Various Degrees of Completion as In Claim 11	9
C. The Invention of the '454 Patent Triggers Execution by Transmitting Data Rather Than Instructions as in Claim 2	11
IV. Asserted Prior Art	11
A. Lindholm '685	11
B. Amanatides	14
C. Selzer	15
V. PETITIONER HAS FAILED TO PROVE THE CLAIMS OF THE '454 PATENT ARE OBVIOUS UNDER ANY GROUND	17
A. Ground 1: Lindholm References	17
1. Neither Lindholm '685 Nor Lindholm '913 Are Prior Art	17
a. The '454 Patent Inventors Conceived of the Inventions At Least by August 24, 2001	18
b. The '454 Patent Inventors Exercised Diligence in Constructively Reducing to Practice	24
i. The '454 Patent Was Constructively Reduced to Practice At Least by November 20, 2003	24
ii. The Inventors and Other ATI Employees Diligently Worked Every Business Day to Reduce the Claimed Invention to Practice	24

c. The Federal Circuit Ruled That the Same R400 Evidence Established Diligence.....	27
d. Petitioner’s Arguments Regarding a Prior ITC Case Not Involving ATI Are Incorrect.....	29
2. No Executing Operations Depending Upon an Amount Of Space Available In The Store (Claim 5), Much Less Performing Vertex Operations Or Pixel Operations Until Enough Storage Is Available For The Other Operation Type (Claims 1, 3, 4)	29
3. No Processor Unit That Performs Vertex Manipulation Operations and Pixel Manipulation Operations At Various Degrees Of Completion Based On Switching Between Instructions In The Instruction Store (Claim 11).....	32
4. No Execution of Instructions “In Response to” Receiving Selected Data (Claim 2).....	33
5. No “Memory” Separate From the “Store” (Claim 6).....	34
6. No “Control Signal” (Claims 7, 10) or “Arbiter” (Claim 10)	35
B. Ground 2: Amanatides + Kohn.....	36
1. No Executing Operations Depending Upon An Amount Of Space Available In The Store (Claim 5), Much Less Performing Vertex Operations Or Pixel Operations Until Enough Storage Is Available For The Other Operation Type (Claims 1, 3, 4)	36
2. No Processor Unit That Performs Vertex Manipulation Operations And Pixel Manipulation Operations At Various Degrees Of Completion Based On Switching Between Instructions In The Instruction Store (Claim 11).....	43
3. No “Selected Data” (Claims 2, 5).....	44
4. Alleged “Selection Circuit” Is Not Within Alleged “Unified Shader” As Required (Claims 7, 10).....	46
5. No “Control Signal” (Claims 7, 10)	49
6. No “Selection Circuit” or “Arbiter,” “(Claim 10).....	50
C. Ground 3: Selzer + Fiske	52
1. No Executing Operations Depending Upon An Amount Of Space Available In The Store (Claim 5), Much Less Performing Vertex	

Operations Or Pixel Operations Until Enough Storage Is Available For The Other Operation Type (Claims 1, 3, 4)	52
2. No Processor Unit That Performs Vertex Manipulation Operations And Pixel Manipulation Operations At Various Degrees Of Completion Based On Switching Between Instructions In The Instruction Store (Claim 11).....	57
3. No Execution of Instructions “In Response to” Receiving Selected Data (Claim 2).....	59
4. Alleged “Sequencer,” “Instruction Store,” “Circuitry,” “Selection Circuit,” and “Arbiter” Are Not Within Alleged “Unified Shader” As Required (Claims 2, 5-7, 10, 11)	60
5. Alleged “Sequencer” And “Instruction Store” Do Not Maintain Instructions (Claims 2, 5, and 11)	63
6. No “Arbiter” (Claim 10) or “Control Signal” (Claims 7, 10)	65
7. No Motivation to Combine Selzer and Fiske	67
D. Secondary Considerations.....	68
1. Initial Skepticism of ATI’s Unified Shader.....	69
2. Unexpected Results in Developing the Unified Shader	70
3. The Unified Shader’s Satisfaction of a Long-Felt Need and Failed Attempts by Others	70
4. Industry Praise of the Unified Shader in the Xenos GPU	71
5. Commercial Success of the Xbox360 Containing ATI’s Unified Shader.	71
6. Adoption by Others	72
VI. CONCLUSION.....	72

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>ATI Techs. ULC v. Iancu</i> , 920 F.3d 1362 (Fed. Cir. 2019)	17,18,28
<i>CAE Screenplates, Inc. v. Heinrich Fiedler GmbH & Co. KG</i> , 224 F.3d 1308 (Fed. Cir. 2000)	35
<i>Catalina Mktg. Int’l v. Coolsavings.com, Inc.</i> , 289 F.3d 801 (Fed. Cir. 2002)	48
<i>In re Magnum Oil Tools Int’l, Ltd.</i> , 829 F.3d 1364 (Fed. Cir. 2016)	60
<i>Perfect Surgical Techniques, Inc. v. Olympus Am., Inc.</i> , 841 F.3d 1004 (Fed. Cir. 2016)	24
<i>Scott v. Koyama</i> , 281 F.3d 1243 (Fed. Cir. 2002)	25
<i>Singh v. Brake</i> , 317 F.3d 1334 (Fed. Cir. 2003)	18
<i>Townsend v. Smith</i> , 36 F.2d 292 (C.C.P.A. 1929)	18

TABLE OF EXHIBITS

Ex. No	Reference Name
EX2001	Declaration of William Mangione-Smith and CV
EX2002	IPR2015-00325, Declaration of Dr Wolfe, Sept. 9, 2015
EX2003	IPR2015-00325, Declaration of Calvin Watson, Sept. 9, 2015
EX2004	IPR2015-00325, Declaration of Lefebvre, Sept. 9, 2015
EX2005	IPR2015-00325,-00326, and-00330, Deposition Transcript of Calvin Watson, November 4, 2015
EX2006	IPR2015-00325, -00326, and -00330, Deposition Transcript of Dr. Wolfe, Nov. 10, 2015
EX2007	IPR2015-00325,-00326, and -00330, Deposition Transcript of Laurent Lefebvre, Nov.13, 2015
EX2008	Gruber et al, R400 Shader Processor 2001, Oct. 9, 2015
EX2009	R400 Document Library Folder History, November 1, 2000 through April 8, 2005
EX2010	R400 Sequencer Specification (Version 0 4), August 14, 2001
EX2011	R400 Sequencer Specification (Version 2 0), September 24, 2001
EX2012	R400 Shader Processor (Version 0 1), Jan. 23, 2001
EX2013	R400 Top Level Specification (Version 0 2), March 11, 2001
EX2014	R400 Program Logs
EX2015	Microsoft Site Visit (Feb. 2003)
EX2016	Log of Exhibits and File Location
EX2017	R400 Program Review Documentation (Feb. 2003)
EX2018	R400 MM Software Status (Feb. 2003)
EX2019	R400 I/O Presentation (Feb. 2003)
EX2020	R400 I/O Presentation (Feb. 2003)
EX2021	R400 August Program Review
EX2022	Executive Review - R400 (Oct. 2002)
EX2023	R400 Area Estimate
EX2024	R400 Executive Review (Sept. 2002)
EX2025	GFIXIP 9x SX Micro-Architecture Specification
EX2026	WD/IA VGT Micro-Architecture Specification
EX2027	GX9 SPI Specification

Ex. No	Reference Name
EX2028	R400 Top Level Specification (Version 0.2)
EX2029	R400 Folder History Log
EX2030	R400 Folder History Log
EX2031	R400 Folder History Log
EX2032	R400 Review PowerPoint
EX2033	R400 Review PowerPoint
EX2034	R400 Development and Documentation
EX2035	Xenos GFX Change History July 2003 - Dec. 2003 Log
EX2036	R400 GFX Change History March 2003 - Dec. 2003
EX2037	R400 Primitive Assembly
EX2038	R400 Primitive Assembly
EX2039	Development Documentation
EX2040	Development Documentation
EX2041	Development Documentation
EX2042	R400 Performance Verification
EX2043	Development Documentation
EX2044	R400 PAD Program Review (Dec. 2002)
EX2045	R400 EMU Test Regress History Log
EX2046	R400 EMU Test Regress Statistics Log
EX2047	R400 Program Review (Dec. 2002)
EX2048	R400 Program Review (Dec. 2002)
EX2049	Device Development Progress
EX2050	Device Development Progress
EX2051	Device Development Progress
EX2052	Device Development Progress
EX2053	R400 Review Status PowerPoint
EX2054	TV Specification
EX2055	R400 Program Review PowerPoint and Test Results
EX2056	Development Documentation
EX2057	R400 PowerPoint
EX2058	R400 Program Review PowerPoint
EX2059	R400 Program Review

Ex. No	Reference Name
EX2060	Block Development Progress
EX2061	R400 Program Review (Dec. 11, 2002)
EX2062	R400 Program Review (Dec. 12, 2002)
EX2063	Development Documentation
EX2064	Development Documentation
EX2065	Development Documentation
EX2066	R400 Technical Documentation
EX2067	R400 Technical Documentation
EX2068	R400 GFX Change History March 2003 to December 2003 Log
EX2069	Ikos 2002 Spreadsheet
EX2070	R400 - Program Review (Oct. 2002)
EX2071	Sqsp Regress Report
EX2072	Xenos Sq Change Log
EX2073	R400 Regress Testing
EX2074	Virtual Logic 3.1 User's Guide
EX2075	Virtual Logic 3.5.5 User's Guide
EX2076	IKOS Virtual Logic 2.1
EX2077	IKON Screenshots
EX2078	IKON Screenshots
EX2079	GFX9 User's Guide
EX2080	US Patent No. 6,897,871
EX2081	Samsung Exynos 5430 Octa SoC
EX2082	Samsung Exynos 3 Quad 3470 Processor Database
EX2083	HW Emulator First Triangle
EX2084	HW Simulator First Triangle
EX2085	R400 Regress Testing Logs
EX2086	R400 Regress Testing Logs
EX2087	R400 IKOS Status
EX2088	R400 Program Review
EX2089	R400 Program Review Documentation
EX2090	PA Check-in History Log
EX2091	RB Check-in History Log

Ex. No	Reference Name
EX2092	SC Check-in History Log
EX2093	SPI Check-in History Log
EX2094	SP Check-in History Log
EX2095	SQ Check-in History Log
EX2096	SX Check-in History Log
EX2097	VGT Check-in History Log
EX2098	R400 File Logs 10/1/2002 – 4/17/2003
EX2099	Netlist Area Sheets
EX2100	IKOS Schematic
EX2101	Stephen Morein Depo Transcript (<i>AMD v. LG</i>) May 25, 2017
EX2102	Laurent Lefebvre Depo Transcript (337-TA-1044) June 28, 2017
EX2103	IPR2015-00325, Declaration of Laurent Lefebvre
EX2104	Development Documentation
EX2105	Development Documentation
EX2106	Development Documentation
EX2107	Development Documentation
EX2108	Development Documentation
EX2109	Development Documentation
EX2110	IPR2015-00325, EX2018 - EX2056
EX2111	IPR2015-00325, EX2057 - EX2071
EX2112	Block Change Logs
EX2113	Andrew Gruber Depo Transcript (<i>AMD v. LG</i>) July 27, 2017
EX2114	IPR2015-00326, EX2001 - EX2002
EX2115	IPR2015-00326, EX2003
EX2116	IPR2015-00326, EX2004
EX2117	IPR2015-00326, EX2005
EX2118	IPR2015-00326, EX2006
EX2119	IPR2015-00326, EX2007 - EX2072
EX2120	IPR2015-00326, EX2073 - EX2118
EX2121	Supplemental Declaration of William Mangione-Smith
EX2122	Supplemental Declaration of Calvin Watson
EX2123	Deposition Transcript of Hanspeter Pfister (02/16/2024)

Ex. No	Reference Name
EX2124	Beyond3D-ATI Xenos: Xbox 360 Graphics Demystified – July 26, 2017
EX2125	Dictionary of Computing 5th Ed (excerpt)
EX2126	Microsoft Computer Dictionary 5th Edition (excerpt)
EX2127	How GPUs Work, David Luebke, Greg Humphreys
EX2128	2006 Microsoft Corporation Annual Report
EX2129	ATI and NVIDIA Proclaim Different Graphics Processors Architecture Goals, Anton Shilov (December 23, 2004)
EX2130	NVIDIA Chief Architect: Unified Pixel and Vertex Pipelines – The Way to Go. NVIDIA Says It Would Make a Chip with Unified Pipes “When it Makes Sense” Anton Shilov (July 11, 2005)
EX2131	FORTUNE: Review of the Xbox 360 – Nov. 17, 2005
EX2132	TECHSPOT – History of the Modern Graphics Processor, Part 3 – The Fall of 3Dfx and The Rise of Two Giants, Graham Singer (April 10, 2013)
EX2133	Working of Xbox 360 – How Xbox 360 Works (February 11, 2010)
EX2134	Xbox 360: the Ars Technica Review, Ben Kuchera (November 30, 2005)
EX2135	Microsoft Xbox dominates a sluggish 2011 gaming market, Nathan Pensky (January 13, 2012)
EX2136	Microsoft’s Xbox 360, Sony’s PS3 – A Hardware Discussion, Anand Lal Shimpi & Derek Wilson (June 24, 2005)
EX2137	Qualcomm History and its GPU (R)evolution – Ryan Shrout (June 22, 2015)
EX2138	Xbox 360 vs. PlayStation 3: The Hardware Throwdown – IGN, Jesse Schedeen (August 26, 2010)
EX2139	Xbox 360 and Playstation 3’s Graphics Card (GPU) Compared – Red Gaming Tech, CrimsonRayne (March 11, 2013)
EX2140	Xbox 360 GPU – NeutralX2 (April 26, 2006)
EX2141	Wikipedia - Adreno
EX2142	Game Consoles: Global Market share 2006-2010 – Statista
EX2143	NVIDIA – Frequently Asked Questions

Ex. No	Reference Name
EX2144	The Xbox 360 Uncloaked – The Real Story Behind Microsoft’s Next-Generation Video Game Console, Dean Takahashi
EX2145	Andrews - Xbox 360 System Architecture
EX2146	Interview with Ken Kutaragi, President of SCEI (3) – “Why I worked with NVIDIA on the PS3”
EX2147	Scott Wasson, “Details of ATI’s Xbox 360 GPU unveiled,” The Tech Report (May 19, 2005)

I. INTRODUCTION

Petitioner Realtek has failed to prove that any of the challenged claims 1-11 of Patent Owner's ATI's U.S. Patent No. 8,760,454 ("the '454 patent") are unpatentable.

First, the Lindholm references are not prior art, which the Institution Decision ("ID") agreed with, and so Ground 1 fails. The conception of the '454 patent's inventions antedates the Lindholm references' critical date, and ATI worked diligently to reduce the inventions to practice, as found by the Federal Circuit as to a parent patent, regarding the same real-world product.

Second, claims 1 and 3-10 require evaluation of storage space in order to determine what type of operations to perform, with various levels of specificity required in the different claims. All three grounds fail to disclose even the broadest formulation, which the ID agreed with as to Ground 3. All three grounds rely on the concept of priority and/or load balancing, in which the system first processes data that has higher priority than other data; it may be that the system has more of a certain type of data to process, causing that to be higher priority, or it may be a pre-set preference. But there is no disclosure in any reference that the amount of storage space for incoming data is evaluated in determining what to choose to process.

Third, claim 11 requires the ability to perform operations at various stages of completion by switching between instructions stored in the instruction store, *i.e.*,

switching between incomplete operations. None of the grounds disclose that functionality.

Fourth, claim 2 requires executing instructions in response to receiving selected data from the data store. All three grounds fail to demonstrate this as well, because the execution they describe is not done in response to receiving data, but rather in response to receiving the instructions, nor is the data “selected” from the data store.

Fifth, the claims require that certain elements are within “unified shader” of the claims, such as the “sequencer” (claims 2, 5), instruction store (claim 11), selection circuit (claim 7), arbiter (claim 10), and others, but Ground 2 and 3 point to items outside the alleged unified shader, and thus the grounds fail as to those claims.

Sixth, claims 7 and 10 require that an arbiter send a control signal to a selection circuit, prompting it to send information to a data store. Petitioner generally fails to identify anything for these elements across the grounds, and what it does point to contradicts a POSITA’s understanding of the terms and its own allegations, for example, when it points to “state data” as a “control signal.”

Lastly, all grounds fail because secondary considerations support non-obviousness of the ’454 Patent.

II. GRAPHICS PROCESSING PRIMER

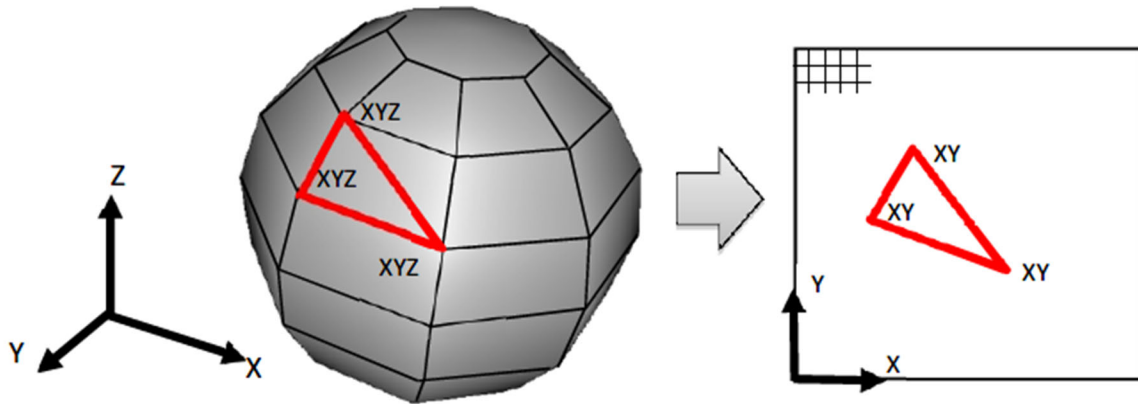
The '454 Patent issued June 24, 2014, and is titled “Graphics Processing Architecture Employing A Unified Shader.” EX1001. The challenged claims concern graphic processors that use a “unified shader” to perform both “vertex” and “pixel” operations. *E.g.*, EX1001 ('454 Patent), 1:32-34, 2:58-61; EX2121 (Mangione-Smith Decl.), ¶15.

A. Graphics Processors Use Vertex and Pixel Data to Create Display Images

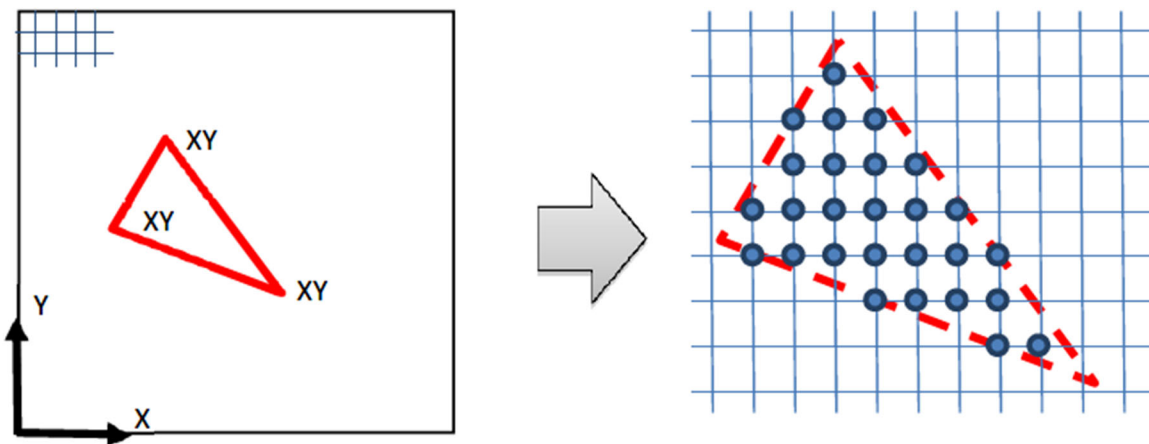
Graphics processors are designed to convert a three-dimensional object into an image for display on a two-dimensional screen. EX2121, ¶16. As part of this process, a three-dimensional object is rendered as multiple shapes, called primitives, such as triangles (see below example). *Id.*



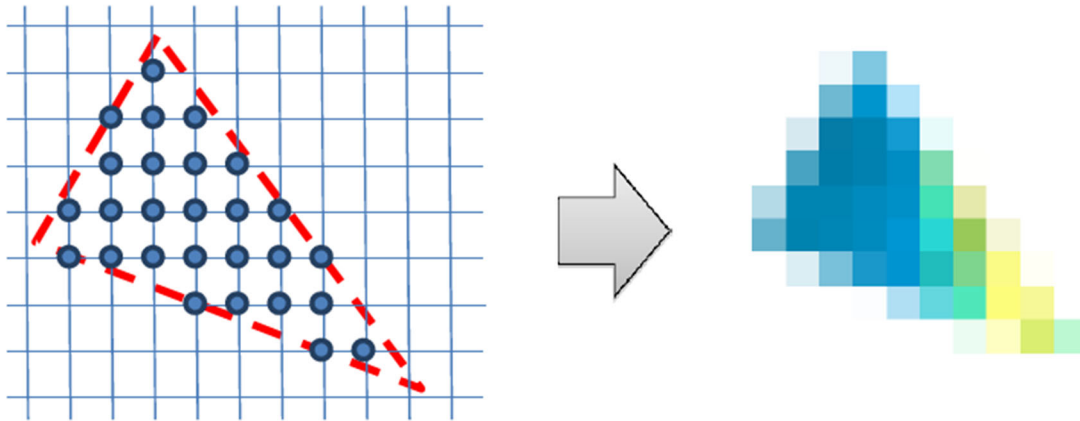
Each primitive has three corner points, or three “vertices.” *Id.*, ¶17. To render a three-dimensional object on a flat screen, each vertex is converted from three-dimensional coordinates to two-dimensional coordinates as illustrated below. *Id.*



Subsequent processing includes rendering each primitive as two-dimensional collection of dots called “pixels.” *Id.*, ¶18. The processor uses the two-dimensional coordinates to determine which pixels fill each primitive (*e.g.*, the XY coordinates depicted in blue below). *Id.*



Further processing includes coloring and applying texture for each pixel filling the primitive. *Id.*, ¶19.



B. Conventional Graphics Processors Use Separate Dedicated Vertex and Pixel Shaders

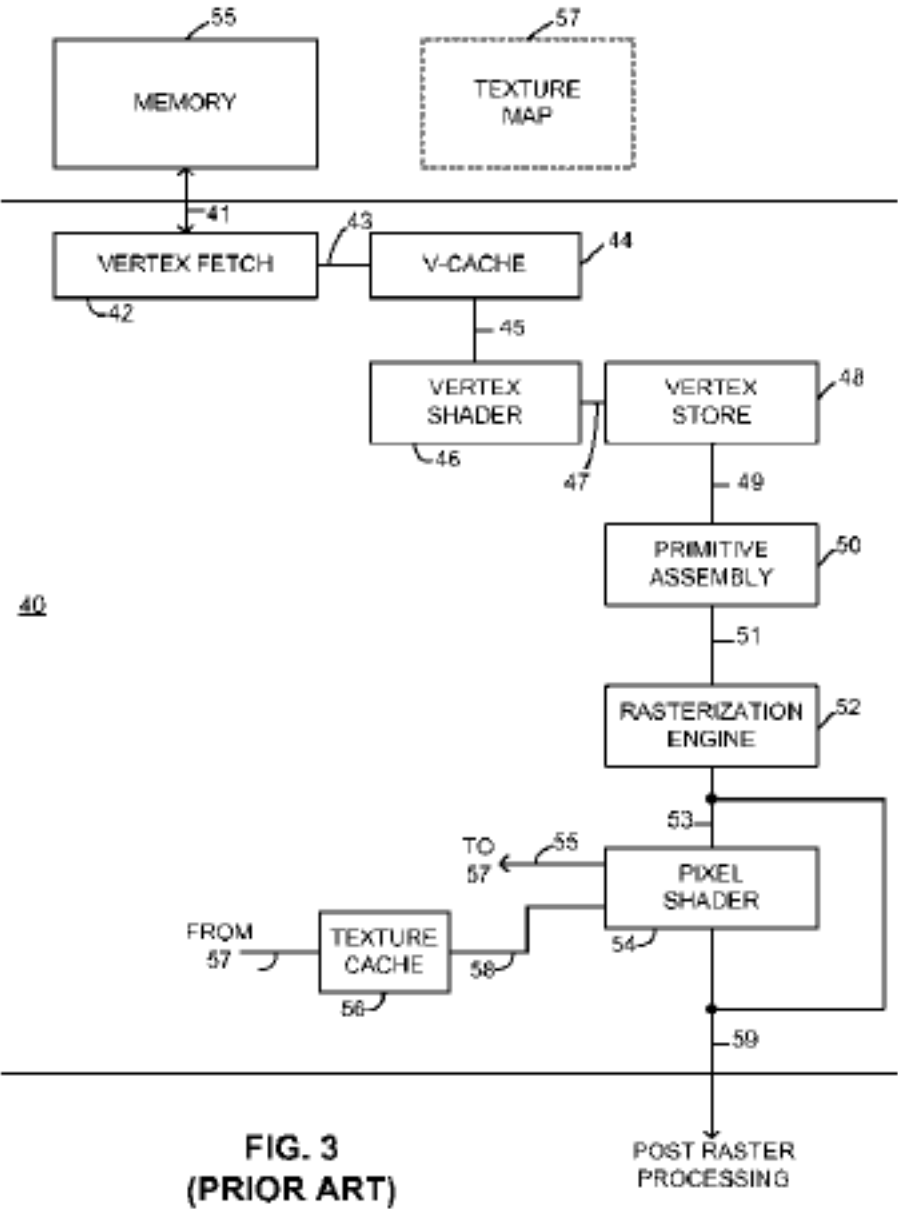
Conventional graphics processors featured a number of circuits, each configured to perform a single type of transformation. These circuits are known as “shaders.” *Id.*, ¶23.

Conventionally, each shader was dedicated to executing a particular type of operation to transform a particular type of data. *Id.*, ¶24. For example, a “vertex shader” would execute only vertex operations to transform vertex data. EX1001, 1:63-2:7. But to execute pixel operations to transform pixel data, another, separate “pixel” shader was needed. *Id.*, 2:8-18. This requirement resulted in drawbacks in both size and performance. EX2121, ¶25.

First, the need for multiple specialized shaders increased the size of graphics processors and overall integrated circuit. EX1001, 2:20-29. Second, sequencing the pixel shader after the vertex shader meant that one shader was dependent on the

output of the preceding shader before it could continue processing, leading to idle downtime and less optimal performance. *Id.*, 2:20-29; EX2121, ¶26.

Such conventional “pipeline” is illustrated below, depicting vertex shader 46 in series with pixel shader 54. *Id.*, ¶27.



EX1001, Fig. 3.

III. '454 PATENT OVERVIEW

Petitioner challenges claims 1-11. Claims 1-5 and 11 are independent claims. Claims 6-10 depend on claim 5.

A. The '454 Patent's Unified Shader Determines Which Data To Process By Evaluating Storage Capacity As Seen In Claims 1, 3-10

In contrast to prior approaches (including that of the asserted prior art), which may use pre-determined priorities or balance workloads based on the amount of work in the queue for execution, the '454 Patent describes determining whether to process a vertex or pixel operation based on available storage space. EX2121, ¶31.

For example, the patent describes an “arbiter circuit for selecting one of a plurality of inputs for processing [by the unified shader] . . . wherein the shader performs one of the vertex operations or pixel operations based on the [arbiter's] selected one of the plurality of inputs.” EX1001, 2:58-3:17; EX2121, ¶32. The arbiter's selection criteria, or “arbitration scheme,” includes at least the capacity to store particular types of data. *Id.* To this end, the data storage (the unified shader's register block) is monitored for available space, which informs what data is selected to be transmitted to the processor to be processed: “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.” *Id.*, 5:32-44.

In use, for example, when there is not enough room in the data storage for vertex data, the processor “continues pixel calculation operations that are to be or are currently being performed [by] the processor based on instructions maintained in an instruction store *until enough registers within the general purpose register block become available*” to store vertex data. EX1001, Abstract (emphasis added); EX2121, ¶33.

This is in contrast with the prior art approach of choosing what to process based on how much of a given type of data is in the queue to be processed, or based on some pre-set priority or other prioritization scheme.

The claims require this functionality in a variety of different ways, with different levels of specificity. *Id.*, ¶34.

Claim 1 is the most specific, requiring performing vertex operations until there is no more room in the general purpose register block for incoming vertex data, in which case continuing to perform pixel operations until enough space appears to once again store vertex data. *Id.* This requires two evaluations: first that there is no longer room for additional vertex data in the general purpose register block (at a time when the system is performing vertex operations), and second, that

at a later point in time (when the system is performing pixel operations), there is once again room for additional vertex data in the general purpose register block. *Id.*

Claims 3 and 4 are broader. *Id.*, ¶35. Claim 3 requires a processor unit to be “operative to perform pixel calculation operations until enough shared resources become available and then use the shared resources to perform vertex calculation operations.” *Id.* Claim 4 requires the opposite; the processor unit must be “operative to perform vertex calculation operations until enough shared resources become available and then use the shared resources to perform pixel calculation operations.” *Id.*

Claim 5 is the broadest, requiring that the processor unit “execute vertex calculation and pixel calculation operations on selected data maintained in a store depending upon an amount of space available in the store.” *Id.*, ¶36.

As explained below, the prior art references fail to teach even the broadest of these formulations.

B. The Unified Shader Can Simultaneously Execute Vertex and Pixel Operations and Switch Quickly Between Operations at Various Degrees of Completion as In Claim 11

Further, the patent describes and claims how to allocate the unified shader’s processing resources among vertex and pixel data in a manner that prevents backlogs of data waiting to be processed and idle downtime for the shader. *Id.* Under the claimed approach, the disclosed unified shader switches back and forth

between unfinished operations to prevent downtime and backlogs of data sitting waiting to be processed. EX1001, 5:23-36; EX2121, ¶37.

To illustrate, when executing a vertex operation, the unified shader may encounter a vertex instruction that it cannot yet perform because the necessary data or memory is not yet available. EX1001, 5:36-44; EX2121, ¶38. Rather than having the unified shader sitting idle, waiting for the necessary data or memory so that it can proceed with the rest of the vertex operation, the unified shader switches to pending pixel operations and executes pixel instructions until the idle vertex operation is ready to proceed. EX1001, 5:32-52.

Thus, the '454 Patent discloses a unified shader that “has the ability to simultaneously perform vertex manipulation operations [vertex operations] and pixel manipulation operations [pixel operations] *at various degrees of completion* by being able to freely switch between such program or instructions very quickly.” *Id.*, 5:32-36 (emphasis added); *see id.*, claim 11; EX2121, ¶39. “[T]hrough 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.” EX1001, 5:49-52. The '454 Patent thus discloses and claims a unified shader that can switch back and forth between executing *unfinished* vertex and pixel operations. *Id.*

C. The Invention of the '454 Patent Triggers Execution by Transmitting Data Rather Than Instructions as in Claim 2

The '454 patent specifically provides the receipt of selected data is what causes the processor to perform the corresponding operations: “the shader performs one of the vertex operations or pixel operations *based on the selected one of the plurality of inputs.*” *Id.*, 2:58-3:17; *see id.*, claim 2; EX2121, ¶¶41-42. This approach is in contrast with the approaches in the asserted references, in which the shader performs tasks *based on the received instructions, not the received data as claimed.* *Id.* This data-driven approach where the receipt of particular selected data dictates the execution of instructions enables the invention’s ability to quickly switch between partially-completed tasks, as the data drives the decision-making about what to process, rather than the instructions doing so. *Id.*

IV. ASSERTED PRIOR ART

A. Lindholm '685

Lindholm '685 regards graphics architecture with a separate pixel input buffer and vertex input buffer, and several execution pipelines:

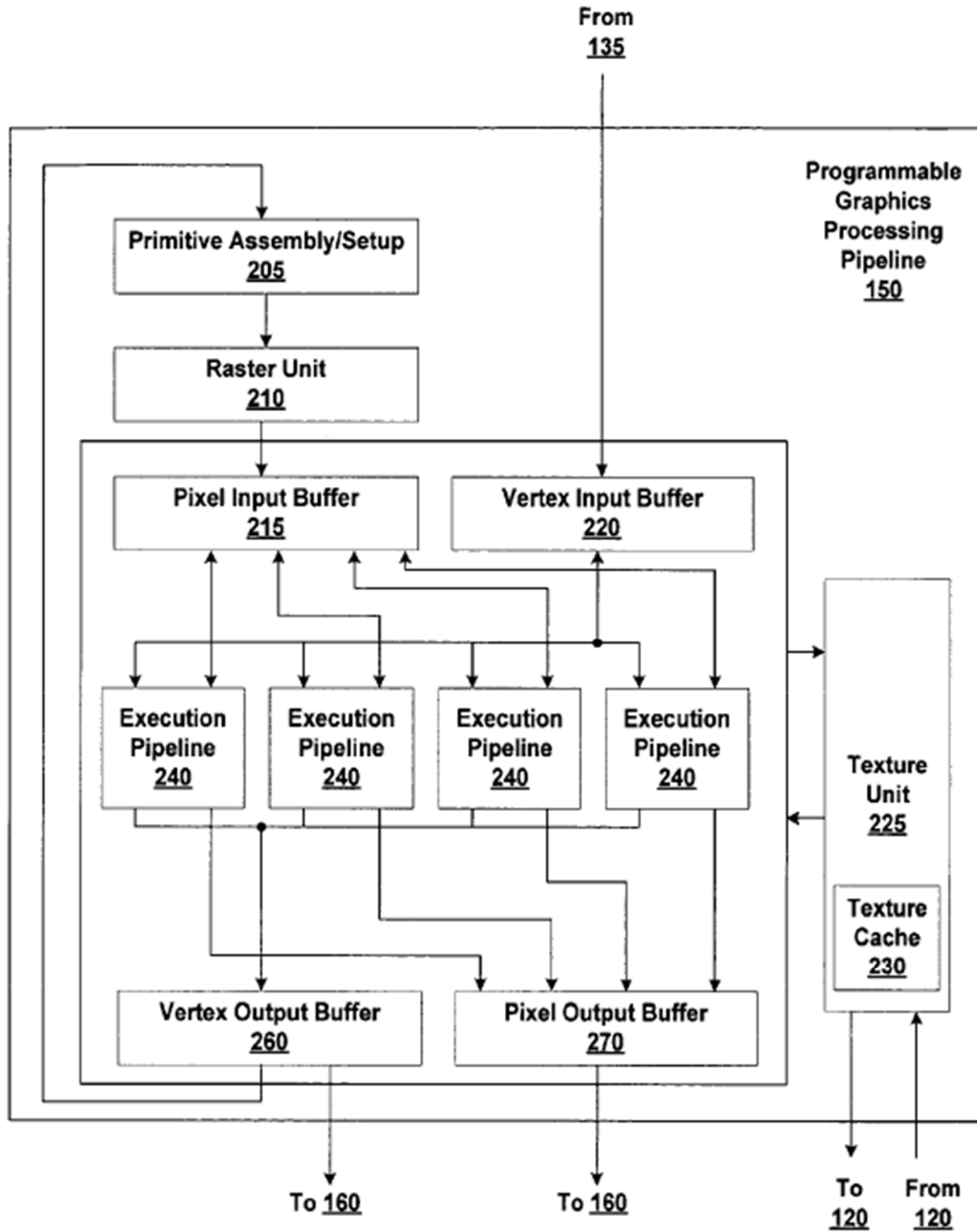


FIG. 2

EX1006 (Lindholm '685), Fig. 2; EX2121, ¶303. An execution pipeline is depicted below:

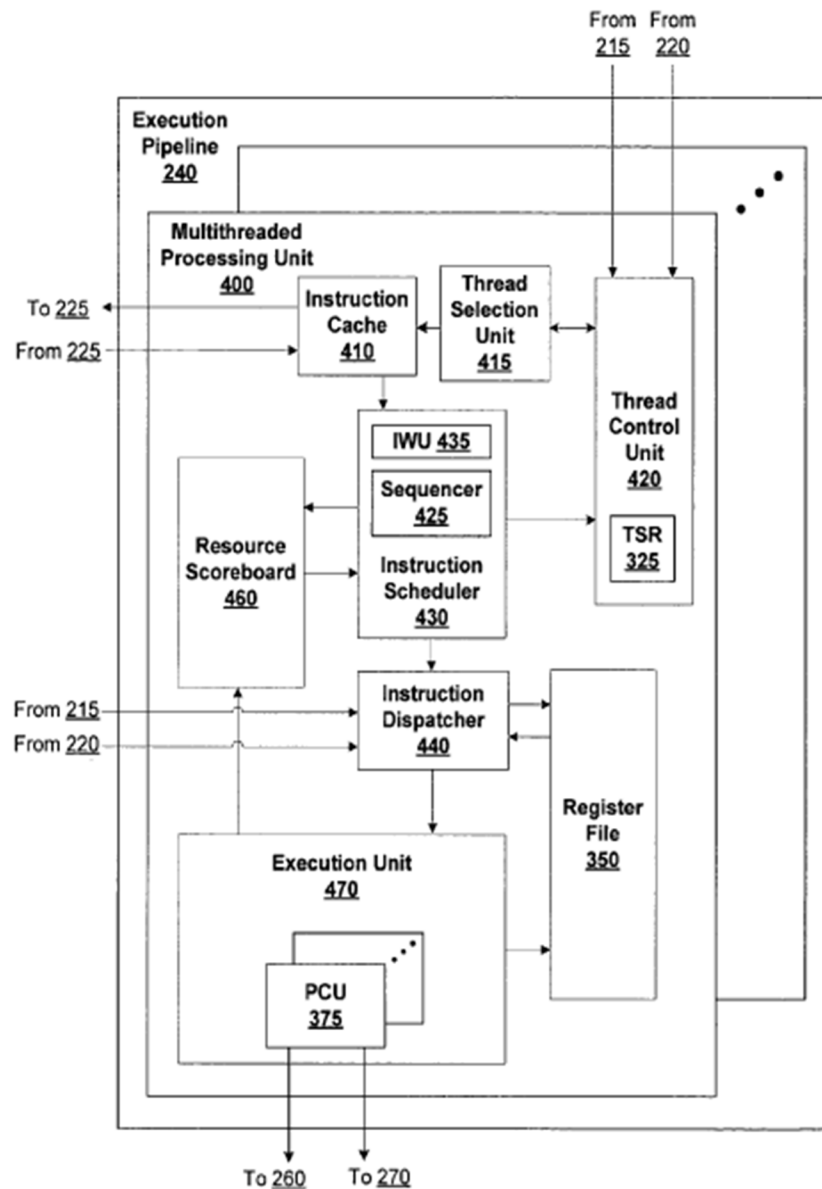


FIG. 4

EX1006, Fig. 4. In response to “program instructions,” “programmable computation units (PCUs) within an Execution Pipeline 240 [] perform operations such as tessellation, perspective correction, texture mapping, shading, blending, and the like.” *Id.*, 5:23-29. “Processed samples are output from each Execution

Pipeline 240 to a Pixel Output Buffer 270,” and then “[t]he processed samples are output from Pixel Output Buffer 270 to Raster Analyzer 160.” *Id.*, 5:30-35. After rasterization, the pixel data is sent to another PCU for pixel processing. *Id.*, Figs, 2, 4, 10:55-11:14.

B. Amanatides

Amanatides regards graphics architecture in which a “host distributes primitives in a round-robin manner” to processors, labeled “G/R”:

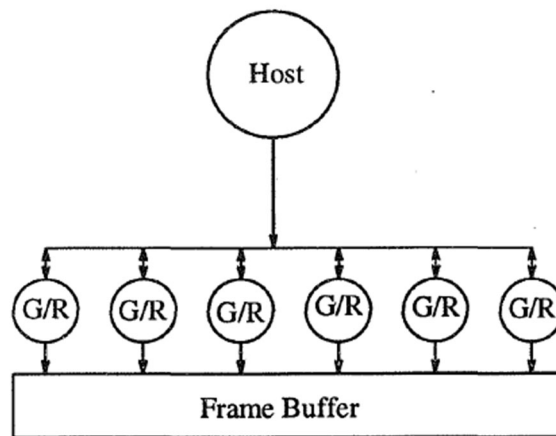


Figure 3

EX1007 (Amanatides), Fig. 3, 157; EX2121, ¶304. “Each processor performs the front-end tasks ... for the primitives. It then broadcasts the results to the other processors to perform the back end operations of rasterization.” EX1007, 157.

The reference also discusses “load balancing” based on “[f]or example, if there is a lot of texture mapping” tasks to be done and based on prioritization of certain types of actions. *Id.*, 157, 159.

C. Selzer

Selzer describes graphics architecture with a traditional pipeline involving a first stage of geometry processing, followed by a second stage of rendering processing:

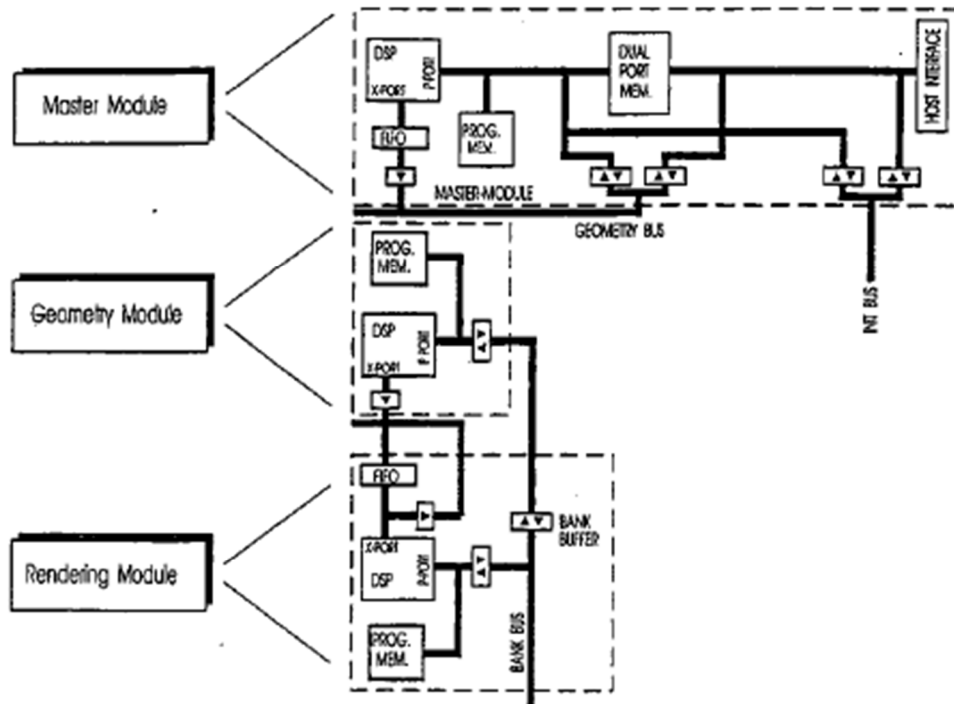


FIGURE 4.2. Basic module structure

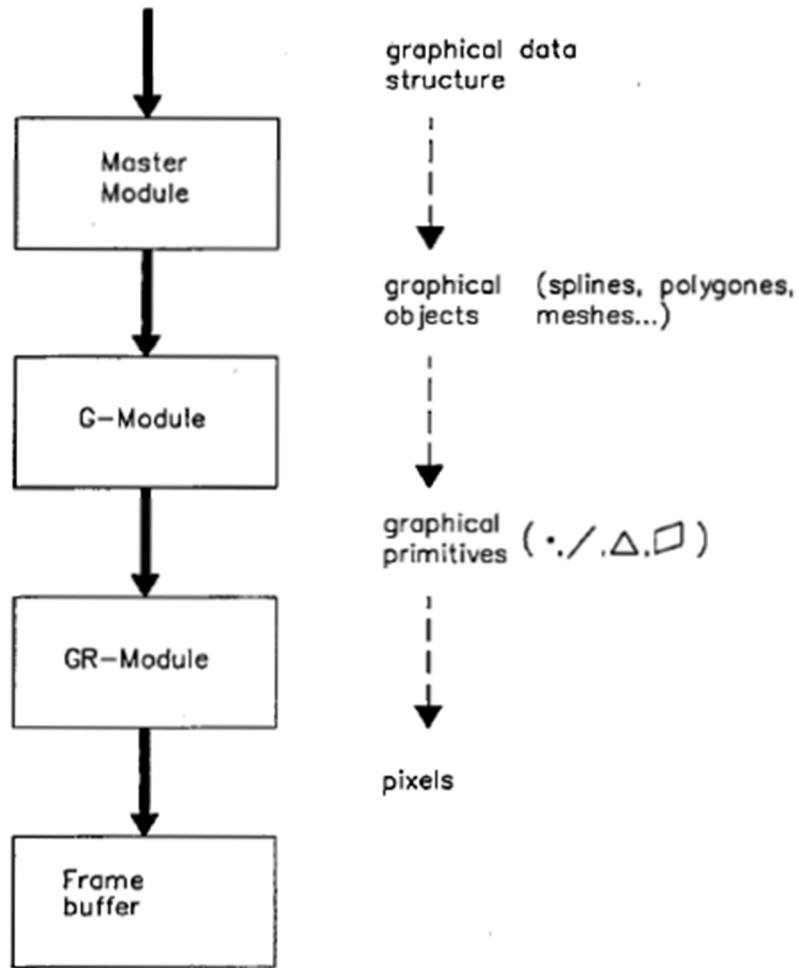


FIGURE 4.5. Graphics data processing

EX1009 (Selzer), Figs. 4.2, 4.4; EX2121, ¶305. “The geometry module performs the transformation, clipping, polygon and patch subdivision, normal interpolation and renormalisation and lighting operations.” EX1009, 41. Alternatively, a rendering module could perform that task. *Id.* After that step, the geometry module or rendering module that performed the geometry processing “deliver[s] processed data to the FIFOs of the appropriate [separate] rendering modules” for rendering. *Id.*

V. PETITIONER HAS FAILED TO PROVE THE CLAIMS OF THE '454 PATENT ARE OBVIOUS UNDER ANY GROUND

A. Ground 1: Lindholm References

1. Neither Lindholm '685 Nor Lindholm '913 Are Prior Art

The Lindholm references are not prior art because the inventions of the '454 Patent were conceived by August 24, 2001 while designing and developing the R400 graphics processor. This is before the effective filing dates of Lindholm '685 and Lindholm '913 (June 30, 2003 and June 27, 2003, respectively). The inventors thereafter diligently reduced to practice until at least the effective filing date of the '454 Patent, November 20, 2003.

Under the applicable pre-AIA first-to-invent standard, references may be antedated by demonstrating: “(1) prior reduction to practice; or (2) prior conception ‘coupled with due diligence.’” *ATI Techs. ULC v. Iancu*, 920 F.3d 1362, 1369 (Fed. Cir. 2019). Here, the evidence demonstrates that the inventors conceived of the invention before the Lindholm references’ critical dates and diligently reduced the invention to practice. Thus, as correctly found in the ID, Patent Owner has antedated the Lindholm references. ID, 33.

The Federal Circuit has already ruled as to the same prior art references and with respect to a parent to the '454 Patent, U.S. Patent No. 6,897,871 (the “'871 Patent”), that (1) “the record is clear that ATI exercised the requisite reasonably continuous diligence” regarding the R400 project, and (2) the R400 project

establishes conception and reduction to practice before the critical date of the Lindholm '913 patent, as to the claims of the '871 patent. *ATI*, 920 F.3d at 1374. The Board correctly found that the Federal Circuit ruling, combined with additional evidence related to the claims of the '454 patent established earlier conception and diligence. ID, 32-33 (considering “relevant aspects of the Federal Circuit’s conclusion that Patent Owner demonstrated sufficient diligence to reduce the invention at issue . . . to practice”).

a. The '454 Patent Inventors Conceived of the Inventions At Least by August 24, 2001

The inventions were conceived of at least by August 24, 2001. EX2121, ¶¶71-194. Conception is the mental formulation and the disclosure of a complete idea for the claimed subject matter. *Townsend v. Smith*, 36 F.2d 292, 295 (C.C.P.A. 1929). Conception is complete when the idea “encompass[es] all limitations of claimed subject matter.” *Singh v. Brake*, 317 F.3d 1334, 1340 (Fed. Cir. 2003).

The following documents demonstrate that the inventors had conceived of the claims by at least August 24, 2001:

1. **“R400 Top Level Specification,”** version 0.2 (**EX2013**), last revised **March 14, 2001** and authored by inventor Steve Morein. This exhibit is authenticated by the Watson Declaration submitted in IPR2015-00325, which was resubmitted here as EX2003. Paragraphs 124-126 of EX2003 authenticate this exhibit, which was numbered EX2041 in IPR2015-00325, as seen on the first page, and here is EX2013. EX2122 (Watson Suppl. Decl.), ¶6. The ID found this exhibit not

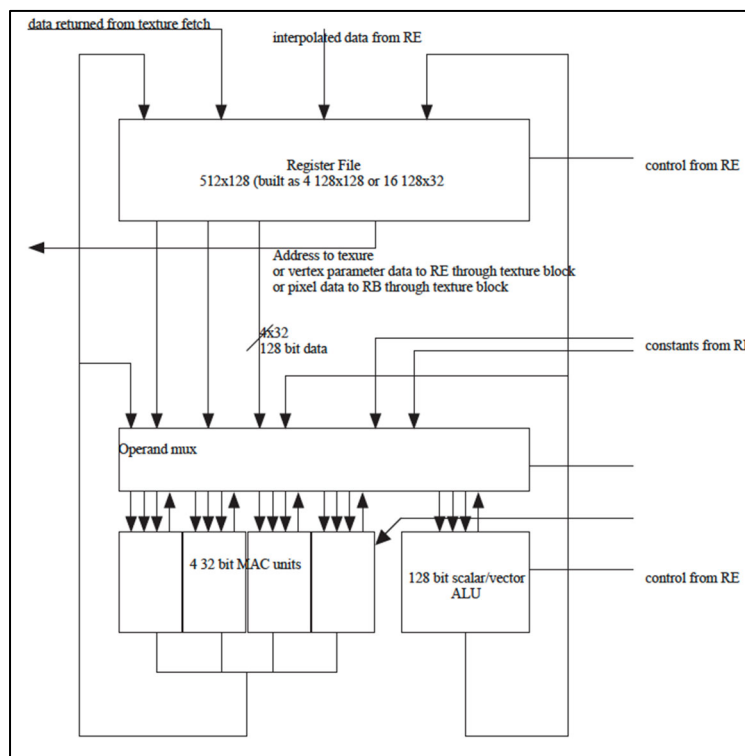
authenticated. ID, 28 n.11. It appears that was due to a confusion regarding exhibit numbering, as it has multiple numbers. The Board held EX2003 (-325 IPR Watson declaration) did not authenticate EX2013 properly. But that declaration used the numbering from IPR2015-00325, which was EX2041, rather than the numbering here, EX2013. Patent Owner respectfully asks that the Board find that EX2003 properly authenticates this document.

2. The R400 “**Shader Processor**” Specification, version 0.1 (**EX2012**), last revised **January 23, 2001** and authored by inventors Andrew Gruber and Andi Skende. This exhibit is authenticated by the Watson Declaration submitted in IPR2015-00325, which was resubmitted here as EX2003. Paragraphs 127-129 of EX2003 authenticate this exhibit, which was numbered EX2042 in IPR2015-00325, as seen on the first page, and here is EX2012. EX2122, ¶6.
3. The “**R400 Sequencer Specification**,” version 0.4 (**EX2010**), last revised **August 24, 2001** and authored by inventor Laurent Lefebvre. This exhibit is authenticated the Watson Declaration submitted in IPR2015-00325, which was resubmitted here as EX2003. Paragraphs 33-35 of EX2003 authenticate this exhibit, which was originally submitted as an exhibit in IPR2015-00325 as Exhibit 2010, the same number as here. EX2122, ¶6.

EX2121, ¶¶72-73.

These documents disclose every element of the claims. *See* EX2121, ¶¶75-194 (element-by-element analysis for claims 1-11). They demonstrate conception no later than August 24, 2001 (the latest revision date), almost two years before the June 2003 dates of the Lindholm patents. EX2121, ¶¶72-73. As an ATI witness stated, the last date in the revision history of the document is the last time it was substantively edited. EX2117 (Watson Decl.), ¶¶14-15; EX2121, ¶73. Although some documents have dates later than August 2001 under “edit date,” that date doesn’t relate to the document’s substance, but is an auto-populated macro which

Specification also explains the R400 implements a unified processing pipe, which includes a processor unit, to perform 3D vertex and pixel operations. EX2013, 9-10; EX2121, ¶¶77, 98, 121, 133, 145, 182. The processor unit of the R400 executes instructions. EX2121, ¶¶114, 166, 171. For example, the unified shader contains vector and scalar ALUs that execute instructions from the instruction store on data in the GPRs (*e.g.*, register file). EX2013, 9-10; EX2011,¹ 11-12; EX2121, ¶¶114, 166, 171. The following graphical depiction demonstrates the structure of the unified processing pipeline. EX2121, ¶¶79, 100, 147.



See EX2013, 31.

The R400 Shader Processor Specification describes the inclusion of general

¹ EX2011 is authenticated by EX2003, ¶¶85-87 as EX2028. EX2122, ¶6.

purpose registers which are configured to store both vertex and pixel data.

EX2121, ¶¶103, 125, 137.

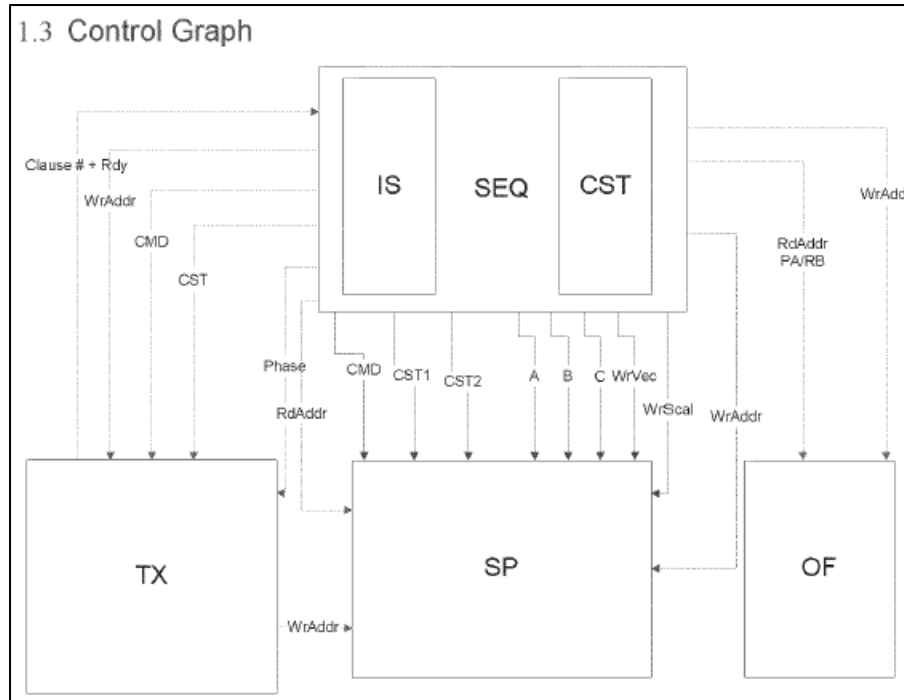
1.1.1 GPR

The general purpose registers are 128 bits wide, composed of four 32 bit values. Depending on the operation these values are interpreted at RGBA, or XYZW, or STQW, or UVQW, or YUVA, or.. to simplify matters the only two aliases used here are XYZW and RGBA.

To hide the latency of memory acceses the shader pipe will switch between different vectors. This is the same as the idea of "microthreading" that some advanced CPU's are investigating. The large register file is split between the vectors executing in the shader pipe. The mangment of the shader register file is automatic, and not visible to a program executing on a vector, execept that a program is required to declare the number of GPRs it need to execute. The hardware will not start a vector until the required number of registers is available. There is a direct tradeoff between the number of registers each program/vector needs and the number of vectors than can be simultainiously resident. If there are too few vectors resident, then the latency of memory acceses can no longer be hidded and performance suffers.

EX2012, 5.

The R400 Top Level Specification and the R400 Sequencer Specification further describe a sequencer having the same characteristics as the sequencer claimed by the '454 Patent. EX2013, 11; EX2010; EX2121, ¶¶76, 82, 85, 92, 105-111. The sequencer, being part of the rasterizer, performs arbitration and is coupled to the general purpose register block and the processor unit. EX2010, 4. The sequencer maintains 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. EX2121, ¶¶105-111, 152.



EX2010, 12. Furthermore, the sequencer “arbitrates between Pixel FIFO and Vertex FIFO – when there are no vertices pending OR *there is no space left in the register files for vertices*, the Pixel FIFO is selected.” EX2010,18; *id.*, 4 (“The sequencer will not start the next vector until the needed space is available.”), 17 (“[T]he arbiter is not going to select a vector to be transformed if the parameter cache is full.”); EX2011, 26 (“There are actually two sets of arbitration -- one for pixels and one for vertices...these threads are further filtered based on whether space is available.”); EX2119, 1281-82, 1284 (same); EX2121, ¶¶73, 83, 127, 139; EX2003, ¶¶33-35, 85-87, 118-120; EX1017, 103:5-21 (R400’s “dynamic mode of the register file allocation method” was “basically functional” and “the basic design was proven to work”). The evidence relied upon by Dr. Mangione-Smith show that ATI conceived of and developed in the R400 features that meet the

claim limitations in the '454 Patent. ID, 26-29 (finding that the R400 Top Level Specification “suggests conception of an idea that was definite and permanent enough that one skilled in the art could understand the invention”). As the Board correctly found, the inventors “conceived of the claimed subject matter in the early stages of R400 development.” *Id.*, 33.

b. The '454 Patent Inventors Exercised Diligence in Constructively Reducing to Practice

i. The '454 Patent Was Constructively Reduced to Practice At Least by November 20, 2003

The '454 Patent claims priority to the '871 Patent, the application for which was filed on November 20, 2003, and supports the claims of the '454 Patent. EX2121, ¶¶245-299 (mapping claims of the '454 Patent to the '871 Patent specification). Thus, the '454 Patent was constructively reduced to practice at least by November 20, 2003. *Id.*

ii. The Inventors and Other ATI Employees Diligently Worked Every Business Day to Reduce the Claimed Invention to Practice

For a constructive reduction to practice, the sufficient diligence period spans from just before the earlier Lindholm's filing date (June 27, 2003) to the inventors' constructive reduction to practice (November 20, 2003). *Perfect Surgical Techniques, Inc. v. Olympus Am., Inc.*, 841 F.3d 1004, 1007 (Fed. Cir. 2016). The evidence actually demonstrates that the inventors exhibited diligence in a much broader period than the requisite period. EX2121, ¶¶195-244, 300-302.

The inventors diligently worked to reduce the R400 to practice, which suffices as diligence towards reduction to practice. *Scott v. Koyama*, 281 F.3d 1243, 1248-49 (Fed. Cir. 2002).

After the R400's conception, the inventors and colleagues worked constantly to implement it. EX2121, ¶¶195-244. This work is shown through records evincing: (i) continuous updates to technical specifications; (ii) development, testing, and debugging of emulation and RTL code, including all critical components, and (iii) synthesis of netlists to be placed on the IKOS system for testing. *Id.*, ¶¶198-199, 227-244. This documentation shows that at least one ATI employee worked every business day during the diligence period to reduce the claimed subject matter to practice. EX2121, ¶¶196, 201; EX2004 (Lefebvre Decl.), 31-56 (showing diligence calendar).

Initially, only a handful of engineers worked on the R400 project. EX2121, ¶196. By late 2001, over one hundred project managers/designers diligently worked to implement and test the R400. *Id.*; EX2113 (Gruber Depo.), 201:22-202:9. In particular, they used the specifications to write emulation code and RTL code for the R400. EX2121, ¶¶200-201.

ATI maintained meticulous records of the development and implementation of the R400 in a revision control system called Perforce, which recorded time stamped logs, user permissions, and files check-outs. *Id.*, ¶¶202-226.

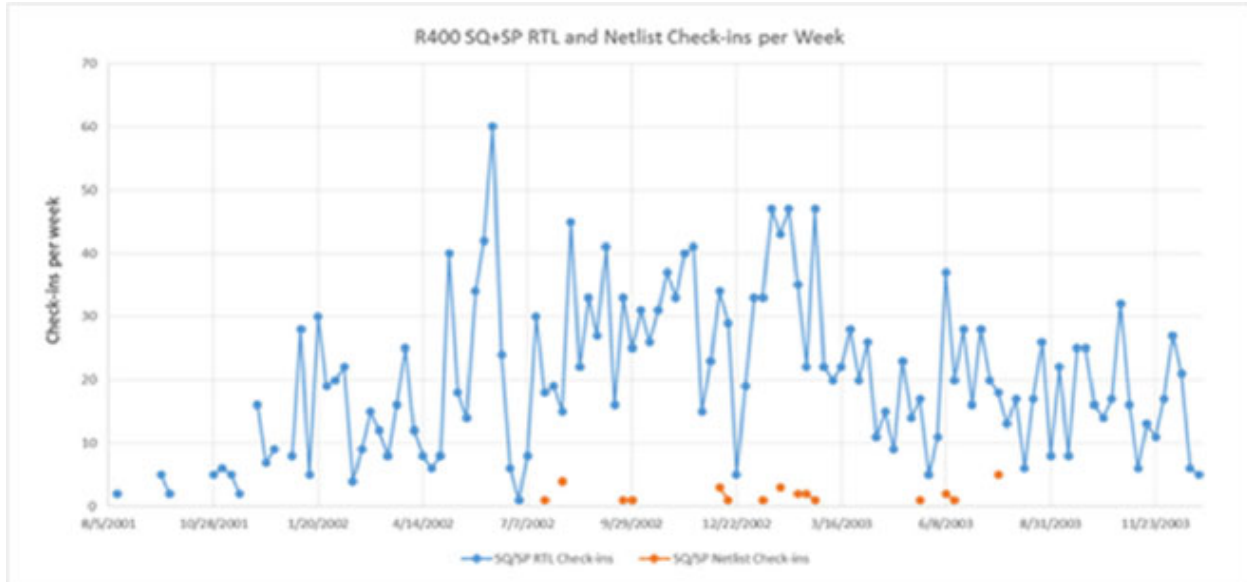
For example, architectural leads of the R400 updated the technical specifications to describe the structure and function of the blocks. EX2121, ¶¶201-202. Some of these specifications were updated approximately every two to three weeks with revisions spanning from May 25, 2001 to August 29, 2003 as indicated by Perforce. EX2117, ¶¶26-129; EX2121, ¶¶201-202; EX2118, ¶¶36, 38-43.

Additionally, the engineers also tracked their efforts in program reviews presentations. EX2121, ¶¶205-207. The program reviews included information regarding the development of deliverables for the R400 project, such as technical specification drafting, emulation code and RTL development (and validation thereof), netlist synthesis, and IKOS testing. *Id.* For example, the following program schedule from the January 17, 2001 presentation captures the critical milestones completed and forecast throughout the development of the R400. *Id.*

Task	Plan	Actual	Forecast
Significant Architecture Issue Identification Complete	10-15-01	10-10-01	
Significant Architectural Issues Resolved	12-17-01	12-20-01	
Emulator Test template Complete	01-18-02		01-18-02
PrimLib has calls for basic tests	02-01-02		02-01-02
Updated Area Estimate Complete / Specs Complete	02-15-02		02-15-02
GC Emulator integration – 1 triangle	02-22-02		02-22-02
Core Emulator pixel / shader tests	03-15-02		03-15-02
Block Testing Begins	04-16-02		04-16-02
GC/Chip Integration Start	05-17-02		05-17-02
Simulate 1 Triangle / Emulator ready for SW	06-03-02		06-03-02
First Syntheses	07-12-02		07-12-02
Verilog Feature Complete	09-16-02		09-16-02
IKOS Emulation start	10-11-02		10-11-02
Synthesis meets Timing	10-25-02		10-25-02
Verilog Frozen	11-08-02		11-08-02
IKOS Emulation (w/ Software) begins	11-11-02		11-11-02
Final Netlist (Gate level ECO only)	11-15-02		11-15-02
A11 Base Layers Tapeout	01-10-03		01-10-03
A11 Metal Layers Tapeout	01-24-03		01-24-03
First Samples for engineering evaluation	03-12-03		03-12-03
A12, A13 Netlist			
A12, A13 Samples			
Volume Ramp			

EX2111 (R400 Program Review, 1/17/2002), 53-69.

Perforce also maintained all changes made to the RTL block level development files. EX2121, ¶¶208. These changes show that continuous diligence was utilized during development of the key architectural blocks incorporated in the R400 from August 24, 2001, to November 20, 2003; only two such blocks (SQ and SP) are detailed in the chart below. *Id.*, ¶¶209-226. The frequency of check-ins for these blocks demonstrates that the R400 team exercised diligence. *Id.*



Id., ¶¶208-212.

c. The Federal Circuit Ruled That the Same R400 Evidence Established Diligence

Considered by the Board as “relevant aspects of the Federal Circuit’s conclusion,” the evidence regarding conception and diligence as to the R400 graphics processor was accepted by the Federal Circuit during IPR of the ’871

Patent, a parent to the '454 Patent, which was challenged under Lindholm '913 in IPR2015-00326. ID, 32-33; *ATI*, 920 F.3d at 1364. In IPR2015-00326, the Board found that ATI established conception, but did not establish diligence in constructively reducing to practice. *Id.*, 1364, 1368.

The Federal Circuit reversed, finding, “[i]nventor Laurent Lefebvre described conception and activities developing the technology in the three Unified Shader Patents. His 60-page Declaration describes his and the other inventors’ and other ATI employees’ activities on this project (designated the R400 project) and was accompanied by ‘almost 1300’ pages of documentary records showing the work done and by whom and when, including metadata, document logs, and folder histories.” *ATI*, 920 F.3d at 1370. Based on this evidence, the Federal Circuit held that the Board erred in holding that diligence had not been shown for the '871 Patent. *Id.*, 1375. The Federal Circuit did not rely on—nor found it necessary to—any code analysis in finding diligence. *Id.*, 1370-72 (relying on inventor declaration and documents, including metadata to conclude diligence).

Here, the evidence demonstrating conception and reduction to practice of the '454 Patent is from the same R400 project that was before the Federal Circuit, and the dates at issue are the same. Moreover, the '454 Patent has similar claims to the '871 Patent. EX2121, ¶246. Therefore, the same result should be found here, as the Board correctly noted. ID, 32-33.

d. Petitioner's Arguments Regarding a Prior ITC Case Not Involving ATI Are Incorrect

Petitioner argues ATI is unable to prove conception the Lindholm '685 patent, June 30, 2003 based on a prior ITC case. Pet., 15 (citing EX1015). This argument is unavailing because that investigation did not address the '454 Patent at all, nor did it involve ATI or any related entity.

Furthermore, the in the ITC case was whether the claims of Lindholm '685 were invalidated by ATI's R400; thus the mapping of the R400 was to the claims of the Lindholm '685 patent, *not* the '454 Patent. EX1015, 141-145.

Finally, the ITC opinion was issued before the Federal Circuit decision discussed above. Even if by some stretch of logic the ITC's holding was relevant here, that holding would have been effectively overruled.

2. No Executing Operations Depending Upon an Amount Of Space Available In The Store (Claim 5), Much Less Performing Vertex Operations Or Pixel Operations Until Enough Storage Is Available For The Other Operation Type (Claims 1, 3, 4)

Claims 1, 3-5 require consideration of the amount of available space for storage impacting what data type to process, as explained above in Section III.A. EX2121, ¶308.

Petitioner's only substantive arguments as to this claim requirement are alleged in the context of claim 3. Pet., 30-35 (claim 3), 35-36 (no substantive

allegations for claim 4), 36-37 (no substantive allegations for claim 5), 40-41 (no substantive allegations for claim 1).

Petitioner has several paragraphs that argue the same thing: priority based on the workload of a given type discloses this claim limitation. *E.g.*, Pet., 31 (“giving priority to allocating vertex threads and pixel threads are only allocated if vertex samples are not available for assignment to a thread”), 32 (priority “based on the amount of pixel samples waiting to be processed”), 32 (prioritization based on “vertex workload or pixel workload, depending on which is greater”), 32 (priority based on “a number of ‘pending’ pixels”). None of these consider the amount of storage, much less the amount of storage for a given type of data, or any of the other more specific requirements regarding what to process based on storage space, as described above in Section III.A. EX2121, ¶310.

Petitioner next points to the statement in Lindholm ’685 that “[e]ach clock cycle, Instruction Scheduler 430 evaluates whether any instruction within the IWU 435 can be executed based on the availability of computation resources in an Execution Unit 470 and source data stored in Register File 350.” Pet., 33. But this evaluates only whether processing resources are free (“the availability of computation resources”) or whether the source data is present such that it is capable of being processed (“source data stored in Register File 350”)—it doesn’t

concern evaluating available storage space for incoming data, much less doing so to determine what to execute. EX2121, ¶311.

Petitioner next alleges that “[t]hread allocation or execution decisions can also be based on space available in Register File 350, or ‘the amount of storage for intermediate data generated during processing of a sample.’” *Id.* This mischaracterizes Lindholm. EX2121, ¶312. All that Lindholm discloses is the common-sense statement that “[t]he maximum number of threads that can be executed simultaneously is related to the number of Execution Pipelines 240, the size of storage for thread state data, the amount of storage for intermediate data generated during processing of a sample” — *i.e.*, if there is more space, more threads can be executed. *Id.* The truism that how many threads can be executed is limited by space doesn’t disclose the required evaluation of available storage space for incoming data to determine what to execute. *Id.* Second, storage capability for intermediate data is irrelevant — it is storage for incoming vertex data (*i.e.*, data to be operated on) that is recited, for example, in claim 1. *Id.*

The ID found this element met because “Lindholm ’685 discloses determining thread execution priority based on the amount of sample data in the pixel and vertex input buffers and recognizes that the maximum number of threads that can be executed simultaneously depends on the size of storage for thread state data.” ID, 40-41. As explained above, neither of those discloses the required

considerations for determining what to process. For example, prioritizing based on how much data is in the buffers doesn't disclose the requirement of claim 1 to perform vertex operations until there is no space for new incoming vertex data, in which case performing only pixel calculations until there is once again room for new vertex data. Nor does it disclose any consideration of space in determining which data type to execute, as also required by claims 3, 4, and 5. EX2121, ¶313.

3. No Processor Unit That Performs Vertex Manipulation Operations and Pixel Manipulation Operations At Various Degrees Of Completion Based On Switching Between Instructions In The Instruction Store (Claim 11)

Claim 11 requires that the “processor unit of the unified shader performs the vertex manipulation operations and pixel manipulation operations at various degrees of completion based on switching between instructions in the instruction store.” EX1001, Claim 11. This requires the processor unit to switch between unfinished vertex operations and pixel operations, as Petitioner's expert agreed. EX2121, ¶324; EX2123, 61:16-62:2.

Petitioner first cites to EX1005, 12:9-27 and EX1006, 7:6-12. These state that if there are not enough threads available, then not all samples within a set of one type can be processed simultaneously. EX1005, 12:14-20. They do not disclose switching between unfinished vertex manipulation operations and pixel manipulation operations at various degrees of completion based on switching between instructions in the instruction store. EX2121, ¶326.

Petitioner next alleges Lindholm discloses “instruction-level scheduling” and the ability to “simultaneously process multiple samples based on Instruction Scheduler and Instruction Dispatcher issuing instructions from multiple threads,” apparently using the word “samples” to correspond to data, as evidence of claim 11. Pet, 34-35. However, the cited disclosures do not discuss switching between different sample types, nor even switching between different samples. EX2121, ¶327. Nor was Dr. Pfister able to identify any disclosures in Lindholm that do. EX2123 (Pfister Dep), 130:14-131:16. Indeed, instruction-level scheduling doesn’t connote switching in the middle of executing a given instruction, which Petitioner’s expert alleged are the “operations” of claim 11. EX2121, ¶¶328-329. The concurrent processing of different samples, which is all Petitioner alleges, doesn’t indicate switching when the processing of a sample is unfinished, as required by claim 11. *Id.*

4. No Execution of Instructions “In Response to” Receiving Selected Data (Claim 2)

Claim 2 requires a unified shader processor that executes certain pixel or vertex thread instructions “in response to” receiving selected data from a data storage, as explained above in Section III.C. EX2121, ¶314. But Petitioner identifies no such disclosure, nor is there any in either of the Lindholm patents.

Petitioner alleges that the “PCU” in Lindholm ’685 perform processes “in response to receiving pixel samples from the Pixel Input Buffer,” citing Lindholm

'685 at 5:23-35. Pet., 28. This citation states that “[p]rogram instructions configure programmable computation units (PCUs) within an Execution Pipeline 240 to perform operations,” which doesn’t indicate that receipt of data causes execution of instructions. EX1005, 5:23-35; EX2121, ¶315. Moreover, Petitioner cites to “thread allocation” to processors as triggering execution, which indicates that threads (groups of instructions), not “selected data,” cause execution of processes. Pet., 30-33; EX1005, 9:5-7.

The embodiments in which locations in Register File 350 are assigned to threads based on priority assigned to each sample type also do not disclose the receipt of data as causing execution of instructions. *See* EX1005, 6:60-7:36; EX2121, ¶316. These disclosures state that threads are assigned to source samples or allocated space in Register File 350. *Id.* The purpose of allocating space in Register File 350 is for “retain[ing] intermediate data generated during execution of program instructions associated with the thread.” EX1005, 7:37-40. Thus, the space allocated in Register File 350 is for data during thread processing, not for data that is to be processed, which is what claim 2 requires. EX2121, ¶316.

5. No “Memory” Separate From the “Store” (Claim 6)

Claim 6, dependent on claim 5, requires “the sequencer further includes circuitry operative to fetch data from a memory.” EX1001, Claim 6. The claimed “memory” must be different from the claimed “store” of claim 5. EX2121, ¶317.

“In the absence of any evidence to the contrary, we must presume that the use of these different terms in the claims connotes different meanings.” *CAE Screenplates, Inc. v. Heinrich Fiedler GmbH & Co. Kg*, 224 F.3d 1308, 1317 (Fed. Cir. 2000).

Petitioner identifies “Instruction Dispatcher 440” as the alleged circuitry. Pet., 37. However, Petitioner identifies the same components for the “memory” as for the “store” of parent claim 5, Pixel Input Buffer, Vertex Input Buffer, and Register File 350, by alleging that Instruction Dispatcher gathers source data from those components. Compare *id.* (identifying “memory”) with Pet., 35 (identifying “store”). This fails to identify a separate claimed “memory.” EX2121, ¶¶318-319.

6. No “Control Signal” (Claims 7, 10) or “Arbiter” (Claim 10)

Claim 7 requires “control signal,” and claim 10 requires an “arbiter,” which provides the “control signal” of claim 7. EX1001, Claims 7, 10; EX2121, ¶320.

Petitioner identifies Thread Control Unit 420 as the claimed “arbiter.” Pet., 37-38. Petitioner doesn’t identify any control signal. *Id.* To the extent it is implying that “thread entries,” “state data,” or “priority” are control signals, that is incorrect. *Id.* Priority information and state data are not control signals. EX2121, ¶321. Control signals are “electrical signals used to control internal or external devices or processes.” EX2126, 128 (Microsoft Computer Dictionary) (emphasis added). This definition is consistent with how the specification describes signals:

“[A]n 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.” EX1001, 5:10-13. Petitioner concedes this, as it argues that the same information satisfies the “information” requirement of claim 7: “Thread Selection Unit output *thread entries* and other thread data, such as state data, which provides *information* to the ‘store.’” *Id.* (emphasis added); EX2121, ¶322. Further, Petitioner hasn’t explained how the alleged control signals are “provided by” the alleged arbiter. *Id.*, ¶323.

B. Ground 2: Amanatides + Kohn

1. No Executing Operations Depending Upon An Amount Of Space Available In The Store (Claim 5), Much Less Performing Vertex Operations Or Pixel Operations Until Enough Storage Is Available For The Other Operation Type (Claims 1, 3, 4)

As explained above in Section III.A, the ’454 patent recites three levels of specificity in terms of ways the claimed invention must evaluate space available in a data storage to determine what operations to perform. EX2121, ¶331. Ground 2 doesn’t disclose any.

Petitioner first points to Amanatides’ “load balancing,” which states “[f]or example, if there is a lot of texture mapping, the processors spend most of their time on this. If most of the primitives are simple, then the processors spend the majority of their time on the front end computation.” EX1007, 157; Pet., 53-54.

Petitioner concludes this “discloses load balancing, namely, performing more vertex operations or more pixel operations depending on . . . *the amount of memory available for each type of work.*” Pet., 54. But this is incorrect on its face. Ex. 2121, ¶332. The citation by Petitioner merely discloses that if there is a lot of one type of work to do, the processors spend more time on that type of work. *Id.* There is no disclosure of considering how much *available* memory space there is for an incoming type of work, as required. *Id.*

Petitioner then points to Amanatides’ high and low priority packets. Pet., 54-55. The high- and low-priority designations are a bus-sharing protocol, in which “[l]ow priority packets get onto the bus only if no high-priority packets are waiting to be sent.” EX1007, 159. As with the above, this doesn’t disclose considering the amount of available storage space. EX2121, ¶333.

The ID relied on the implementation of the priority scheme in Amanatides to find it discloses the above elements. ID, 57. However, in Amanatides there are only ever three decision points regarding what data to transmit and process. EX2121, ¶334. None of them involve deciding what type of packet to execute even in part based on how much space is available in the input FIFOs, the alleged “general purpose register block” (claim 1), “shared resources” (claims 3, 4), and “store” (claim 5)). *Id.* Petitioner also alleges a few other items as comprising the

register block, but notably doesn't allege that the output FIFO is a part of the claimed register block, although the correct conclusion would be unchanged. *Id.*

First, in this scheme, GMs (the alleged vertex data and/or the messages that correspond to vertex operations) are sent for processing. EX1007, 160; Pet., 51, 54-55; EX2121, ¶335. This first decision point (to the extent any decision is made at all during startup) doesn't involve a determination of available space in the input FIFOs, or any other alleged component that Petitioner alleges comprise the general-purpose register block, shared resources, or store. *Id.* This is simply the default starting rule. EX2123, 38:23-39:6 (Petitioner's expert agreeing that at startup, GMs are sent from the host, and there is no disclosure of the host sending RMs).

Second, "[a]s soon as one of these GMs turns into a RM it shuts down further introduction of GMs from the host," triggering a second phase of operation. EX1007, 160; EX2121, ¶336. Thus, the second decision point, the transition from sending GMs for processing to sending only RMs for processing doesn't consider available storage space in the input FIFO or any other alleged component that Petitioner alleges comprise the general-purpose register block, shared resources, or store. *Id.* The only consideration is that one GM has been turned into an RM. *Id.* Petitioner's expert conceded that fact. EX2123, 40:15-44:18. After that triggering event, in this second phase of operation, only RMs are sent to input FIFOs for

processing, and they are sent from the output FIFOs that store data after operation.

EX1007, 159-160. Therefore, to the extent that the ID held that the statement “[w]henver space appears at any input FIFO it is the RMs that are delivered to it” indicates an evaluation of available storage space in determining which data type to process, that is incorrect, because there is no determination of data type to process in this phase: once one GM is converted, only RMs are sent and processed.

Third, a third phase of operation, in which “the host begin to deliver GMs again” is triggered by the system running out of RMs to deliver to input FIFOs for operation, *i.e.* “when all the output FIFOs are empty and the processors are working on the remaining RMs in their input FIFOs.” *Id.*, 160; EX2121, ¶¶337-338. This third decision point, the transition from sending only RMs for processing to sending GMs for processing, therefore also doesn’t consider available storage space in the input FIFO or any other alleged component that Petitioner alleges comprise the general-purpose register block, shared resources, or store, as Petitioner’s expert conceded. *Id.*; EX2123, 47:11-48:5.

The ID appears to hold that this transition point considers storage space in the output FIFO. *ID*, 57. That is incorrect, because it merely evaluates whether there is RM work to do, not whether there is finally once again enough storage space for GMs, which it never determined was lacking in the first place. Moreover, GMs are never delivered to the output FIFOs, so evaluation of space

there would not establish there is space for storing incoming GMs as required, anyway. And even if the ID were correct, the output FIFO is not part of the alleged “general purpose register” (claim 1), “shared resources” (claims 3, 4), and “store” (claim 5). EX2121, ¶338. The input FIFO is alleged to fill that role, among other components, such as DRAM, a bus, and an on-chip cache. Pet., 46-47. And the output FIFO could not fill the role of the general-purpose register block, as that is the memory in the claims that data is put for processing (which Petitioner contends is the input FIFO), whereas the output FIFO is the place for data after processing. EX2121, ¶338.

Thus there is no disclosure in Amanatides that meets even the broadest recitation of the concept, which is in claim 5. *Id.*, ¶339.

Amanatides further fails to disclose the narrower requirements of claims 1, 3, and 4 as well. *Id.*, ¶340.

As to claim 1, there is no consideration in Amanatides of whether the alleged “general purpose register block” (the input FIFO and/or other components) has enough available space to store the alleged incoming vertex data (GMs). *Id.* In the first phase of operation, GMs are sent, stored and operated on; and nothing in Amanatides causes the system not to send and process GMs if “the general purpose register block does not have enough available space therein to store incoming vertex data.” *Id.* In the second phase, there very well might be space to store GMs

in the input FIFOs. *Id.* But because the output FIFOs still have RMs, the system refuses to attempt to send, store, and process any GMs. *Id.* It chooses not to store GMs *not* because it cannot fit them in memory, as required by the plain claim language, but because it prefers to work on RMs until all RMs are gone to prevent deadlock. *Id.* And there is no disclosure of processing RMs “until enough registers within the general purpose register block become available,” as required in claim 1. *Id.* There is simply no evaluation of whether to cease processing RMs, or begin sending and processing GMs, on the basis that there is once again enough space for GMs in the input buffers or anywhere else. *Id.*

Similarly, as to claim 3, the system in Amanatides doesn’t perform the alleged pixel calculations (RMs) “until enough shared resources become available” for the alleged vertex calculations (GMs). *Id.*, ¶341. Rather, it performs RM processing until it runs out of RMs to process. *Id.* To the extent the space of anything is evaluated, it is the output FIFO, which are not the alleged “shared resources,” which are the input FIFO. Pet., 46-48, 56-57 (alleging “input FIFO” but not the output FIFO as the GPRB). However, even that is not an evaluation of whether enough shared resources are available: it simply states that there is no more RMs to process. EX2121, ¶341. Enough resources may have been available before that time, or they still may not be available when there are no more RMs to process; Amanatides simply doesn’t check. *Id.*

Nor does Amanatides disclose claim 4. EX2121, ¶342. Even if the Board disagrees with all of the above, Amanatides cannot be said to perform alleged vertex processing (on GMs) until enough shared resources for the alleged pixel processing (on RMs) becomes available as required by claim 4, which requires the reverse sort of operation of claim 3. *Id.* The only transition from GM processing to RM processing is explicitly “[a]s soon as one of these GMs turns into a RM.” EX1007, 160. Even if petitioner were right about the transition from RM processing to GM processing, which is not, as explained above, the only evaluation of when to begin performing pixel processing (allegedly the RM processing), as required by claim 4 has nothing to do with storage space. EX2121, ¶342.

Claim 5 is not disclosed for an additional reason. *Id.*, ¶343. Petitioner doesn’t specifically allege what is the “store.” Pet., 57. The ID incorrectly states Petitioner alleges the input FIFO is the “store.” ID, 57. Instead, for claim 6, Petitioner claims the “FIFOs” are the “selection circuit” that sends packets *to the store*, and thus cannot be the “store.” Pet., 59. Thus, Petitioner hasn’t alleged a theory in which the input FIFO is the “store,” and so its allegations fail as to that claim for that reason alone. And what is alleged to be the “store,” the on-chip data cache, is not ever evaluated for storage space (or anything else) to determine what data to transmit and process in Amanatides. EX2121, ¶343. Thus, even if the

Board disagrees with the above analysis regarding claim 5 that is similar to that of the other claims, claim 5 is not invalid for this additional reason.

Kohn is not alleged to fix the deficiency with Amanatides, but rather is used for the concept of switching “at various stages of completion.” *Id.*, ¶344; Pet., 55-56.

Thus, Ground 2 fails to disclose the above limitations of claims 1 and 3-5.

2. No Processor Unit That Performs Vertex Manipulation Operations And Pixel Manipulation Operations At Various Degrees Of Completion Based On Switching Between Instructions In The Instruction Store (Claim 11)

As Petitioner’s expert conceded, this claim element requires switching between partially-complete (*i.e.*, unfinished) operations. EX2123, 61:16-62:2; EX2121, ¶345.

Petitioner argues that the “dual-operation mode” within the “pipelined mode” in Kohn discloses this limitation. Pet., 55-56; EX2121, ¶¶346-347. But neither it nor its expert has done nothing whatsoever to prove that fact other than to alleged without support that “that the functional units within each i860 processor could not only be performing vertex operations and pixel operations simultaneously, but at various degrees of completion, based on switching between instructions in the instruction cache.” Pet., 55.

A pipeline doesn’t operate perform operations at various stages of completion based on switching between instructions in the instruction store.

EX2121, ¶348. In a pipeline, data comes in, one operation is done on it, then it moves to the next stage where another operation is done on it, etc. *Id.* Kohn describes distinct operations in its pipeline: “The RISC core contains a pipeline consisting of four stages: fetch, decode, execute, and write.” EX1008, 17. Thus, there is no support for the argument that Kohn teaches operating them at various degrees of completion, much less based on switching instructions that are in an instruction store. EX2121, ¶348. The instructions are already loaded in the blocks that perform the instruction; there is no fetching of instructions from the instruction store, as required. *Id.* The fact that there might be parallel paths or “dual-operation” doesn’t change that, because it doesn’t change the fact that operations are not performed in a partially-complete state. *Id.*

Moreover, there is no discussion of why using the pipeline of Kohn would be in any way relevant or beneficial to Amanatides, including the relied-upon load-balancing and deadlock solution, or any other part of Amanatides, or even how the proposed combination would work. Pet., 53-56; EX2121, ¶348. Even if some concoction could be made, Petitioner hasn’t adequately explained it or why it would be obvious and motivated.

3. No “Selected Data” (Claims 2, 5)

Claims 2, 5 of the ’454 Patent require selecting data from a data store, transmitting that data to the processor, and operating on that data. EX1001, Claims

2-5. But there is no process in Amanatides or Kohn for selecting data from a data store. EX2121, ¶350. As Petitioner’s expert conceded, to meet this claim limitation, the data must be in the general-purpose register block (claim 2) or data store (claim 5) while it is selected. EX2123, 79:16-24.

But the reliance by Petitioner and the ID on the “priority” of certain data over others in Amanatides doesn’t establish a means by which data is selected “from the general purpose register block” (claim 2) or from “a store” (claim 5). EX2121, ¶351.

There is no selection of GMs versus RMs sitting in any alleged register block or data store. EX2121, ¶352. Rather, as explained in the previous section, there are different modes of operation: at first GM only, then RM only, then GM only. *Id.* The system then “oscillate[s]” back and forth between these modes. EX1007, 160. This doesn’t disclose selecting GM versus RM from a register block or data store. EX2121, ¶352. In the GM-only phases, GMs are sent from the host to the input FIFOs. *Id.* The host is not alleged to be a register block or data store, nor could it be, because, as explained below in the following section, it is outside the alleged unified shader. EX2121, ¶351. In this phase, after being sent from the host, there is no selection; the data is simply processed. *Id.*, ¶352. And RMs, which are only sent to be processed in the RM-only phase, are never sent from the host, in any phase; instead, as Petitioner’s expert conceded, all RMs in the system are

generated only from the act processing GMs, and all of them are delivered then to an output FIFO (which is not alleged to be part of the register block or data store, see Section V.B.1) and then to an input FIFO for processing; there is no selection from the claimed data stores. EX1007, 159-160; EX2123, 35:7-36:4 (agreeing that Amanatides doesn't disclose RMs "com[ing] from any other source besides being converted from GMs").

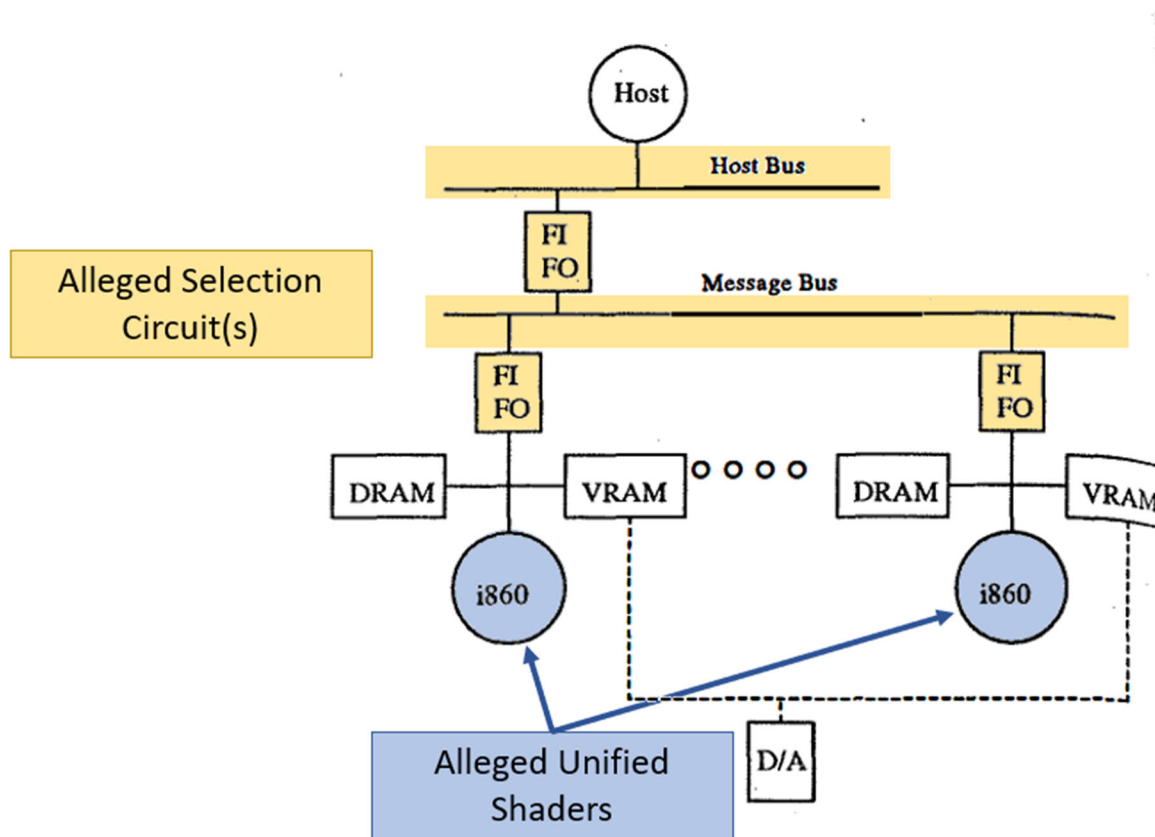
Thus, there is no world in which GMs and RMs are sitting in a register block or data store and are selected from, which is the apparent read by Petitioner and the ID. EX2121, ¶352.

4. Alleged "Selection Circuit" Is Not Within Alleged "Unified Shader" As Required (Claims 7, 10)

Claims 7 and 10 require the "shader of claim 5, further including a selection circuit." EX1001, Claims 7, 10. But the alleged unified shader doesn't include the alleged selection circuit, and thus the ground fails for claims 7 and 10. EX2121, ¶353.

Petitioner alleges the unified shader is each G/R processor, which is equivalent to each i860 processor, *e.g.*, in Figure 4, as Petitioner's expert agreed. EX1007, 158; Pet., 46 ("Thus, Amanatides termed the unified shader unit the G/R unit."), 48 (referencing "each G/R shown in Amanatides (the i860 processor)."); EX2123, 6:2-17, 10:5-11 ("[T]he G/R unit is the i860.").

Petitioner then alleges that the selection circuit is “met by at least *the bus and FIFOs*, which are operative to provide packets (‘information’) to the DRAM, VRAM, and on-chip data cache (‘store’).” Pet., 59. But the bus and FIFOs are not part of the alleged unified shader. EX2121, ¶355. Figure 4 of Amanatides shows that the bus and FIFO are outside of the i860 processors. *Id.*



EX1007, Fig. 4 (annotated). The text of Amanatides explains this is also the case: “The processors would be connected to each other and the host via a bidirectional FIFO connected to a message passing bus.” *Id.* Thus, while the bus and FIFOs are

used to connect the processors, they are not part of each G/R processor alleged to be the unified shader. EX2121, ¶355.

The preambles, each of which recite a unified shader, are limiting, and thus the structure recited within is required. EX1001, Claims 1-11. “[A] preamble limits the invention if it recites essential structure or steps, or if it is necessary to give life, meaning, and vitality to the claim.” *Catalina Mktg. Int’l v. Coolsavings.com, Inc.*, 289 F.3d 801, 808 (Fed. Cir. 2002). The “unified shader” language gives additional structure which has been “underscored as important by the specification,” and is thus limiting. *Id.* The specification explains that the unified shader is, for example, “capable of performing both the vertex operations and the pixel operations in a space saving and computationally efficient manner.” EX1001, 2:58-61; EX2121, ¶356.

Further, claims 1 and 11 also recite “the unified shader” in the body of the claims, making it unambiguous that the term “a unified shader,” which provides antecedent basis in the preamble, is limiting. *Id.* The same is true of other claims, as there is no reason to differentiate them. *Id.*

“Clear reliance on the preamble during prosecution to distinguish the claimed invention from the prior art transforms the preamble into a claim limitation because such reliance indicates use of the preamble to define, in part, the claimed invention.” *Catalina*, 289 F.3d at 808. This was so here, as the applicant

distinguished prior art as having “separate vertex shader and separate pixel shader[s],” arguing there was thus “no teaching or suggestion of any unified shader.” EX1002, 358-359 EX2121, ¶357.

Petitioner doesn’t contest that the preambles are limiting; no claim construction was proposed by Petitioner, and Petitioner sought to prove every preamble was disclosed by the prior art.

5. No “Control Signal” (Claims 7, 10)

Claim 7 requires a “control signal” and claim 10 requires the arbiter to provide the control signal to the selection circuit. EX1001, Claims 7, 10; EX2121, ¶358. Petitioner argues that the arbiter sends GM and RM packets to the selection circuit, and the claimed control signal is part of the packet, in the form a priority designation. Pet., 59-60.

First, there is no disclosure that the packets themselves contain a priority designation. EX2121, ¶359. Petitioner’s theory is that GMs are given low priority and RMs are given high priority. A POSITA would understand that given this pre-set priority scheme, it would be a waste of resources to list the priority in the packet. *Id.* Rather, the system would simply treat the different packets differently based on type. *Id.*

Moreover, neither priority data, or packet type are a “control signal.” As The GM and RM packets are *data*. EX1007, 159. This is consistent with a

POSITA's understanding of packets, which are units of information or data being transmitted. EX2125, 244 (Dictionary of Computing); EX2121, ¶358.

In contrast, a POSITA would understand control signals are “electrical signals used to control internal or external devices or processes.” EX2126, 128; EX2121, ¶360. This definition is consistent with how the '454 Patent's specification describes signals: “[A]n 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.” EX1001, 5:10-13. A control signal is an instruction from the sender to the recipient, not mere data that the recipient can interpret however it chooses. EX2121, ¶360.

6. No “Selection Circuit” or “Arbiter,” “(Claim 10)”

Claim 7 requires “a selection circuit operative to provide information to the store in response to a control signal.” Claim 10 adds the requirement of an arbiter that provides the control signal to the selection circuit. EX2121, ¶361.

Petitioner alleges that “the ‘selection circuit’ of claim 7 is met by at least the bus and FIFOs, which are operative to provide packets (‘information’) to the DRAM, VRAM, and on-chip data cache (‘store’) in response to a control signal, such as whether the packets are designated high-priority or low priority as explained further below.” Pet., 59. But neither the bus nor the FIFOs can be

“selection circuits.” EX2121, ¶362. FIFOs are merely memory, not selection circuits. *Id.* And bus lines are merely transmission lines, not selection circuits. *Id.*

Further, Petitioner alleges that the bus and FIFOs provide packets to the DRAM, VRAM and on-chip cache to meet the claim requirement that the selection circuit be “operative to provide information to the store in response to a control signal.” Pet., 59. But it provides no evidence or analysis—all Petitioner alleges is that the bus is used to transmit packets. EX2121, ¶363. There is no indication in Amanatides that “packets” are ever sent from the FIFOs or bus to DRAM, VRAM, or on-chip cache. *Id.* Moreover, there is no indication or analysis proving that the alleged “control signal” (which is alleged to be a part of the packet itself) prompts such a transmission, as required. *Id.*

Petitioner’s mapping of the “arbiter” also makes no sense. EX2121, ¶364. Petitioner admits the references of Ground 2 lack an arbiter and instead alleges that time-slot arbitration “would be performed by logic circuits.” Pet., 59-60. Petitioner’s expert explained that the theory is that the identified selection circuits make the decision regarding what data type to transmit, and then the undisclosed arbiter is used to put the packets on the bus lines in proper time slots. EX2123, 83:4-8. This mirrors the Petition, which recites that the “arbitration” performed by the arbiter regards time-slot allocation only. Pet., 97. But in this system, the alleged arbitration is separate from and happens *after* selection; the decision is

made and then the final sorting onto the bus via timeslots happens by action of the undisclosed arbiter, as a POSITA would know. EX2123, 82:6-83:8; EX2121, ¶365. At a minimum, that is the most obvious way to implement the arbiter, and without question there can be no inherency argument that it *must* be done in the opposite order, as would be required for Petitioner’s theory. *Id.* In that hypothetical system, there is no “control signal” sent from the undisclosed arbiter to the alleged selection circuit that prompts transmission of information from the selection circuit to the store. *Id.* Instead, the undisclosed arbiter merely receives packets from the selection circuit, which has already done the selection (not based on any control signal from the arbiter), and the arbiter puts it on a bus line. *Id.*

C. Ground 3: Selzer + Fiske

1. No Executing Operations Depending Upon An Amount Of Space Available In The Store (Claim 5), Much Less Performing Vertex Operations Or Pixel Operations Until Enough Storage Is Available For The Other Operation Type (Claims 1, 3, 4)

As explained above in Section III.A, the ’454 patent recites three levels of specificity in terms of ways the claimed invention must evaluate space available in a data storage to determine what operations to perform. EX2121, ¶367. Ground 3 doesn’t disclose any of these formulations.

Petitioner relies on the alleged ability of Selzer to “load balance” as prioritizing vertex or pixel operations based on “processing load,” *i.e.*, how much

work of a given type there is to do. Pet., 73. Selzer explains it prefers that its rendering module work on rendering calculations, and switches only to geometry calculations when the rendering module's input buffer is empty, as this approach operates at "peak performance." EX1009, 47-48. This priority scheme doesn't disclose claims 1, 3, 4, or 5, as it doesn't evaluate available storage space in determining what to process. EX2121, ¶368.

Petitioner also points to thread prioritization discussion in Fiske. Fiske regards a "thread prioritization" algorithm, as stated in its title. EX2121, ¶369. "Thread prioritization is a simple means of guiding the scheduling of threads on a node. Each thread has a priority that indicates the importance of the thread in the overall problem . . . Thread prioritization involves assigning a priority to all the different threads in an application, and then using this priority to make thread scheduling decisions." EX1010, 210-211. There is no assessment of available space for storage that impacts Fiske's priority scheme as required by the claims. EX2121, ¶369.

As explained in Grounds 1 and 2, see Sections V.A.2 and V.B.1, assigning priorities to threads, data, or data types, doesn't disclose determining what data to transmit and process based on an evaluation of current data storage capacity, as required by claims 1 and 3-10 of the '454 Patent. EX1009, 46-47 (rendering module performing geometry processing once it performs all rendering); EX1010,

211 (“Thread prioritization involves assigning a priority to all the different threads in an application, and then using this priority to make thread scheduling decisions. . . . [T]he hardware scheduler uses the priority to determine which loaded thread to execute.”); EX2121, ¶370. Although Petitioner asserts that Selzer and/or Fiske teaches more, such as (in Petitioner’s words) “if enough shared resources are not available for pixel processing, then the Selzer/Fiske combination would allocate its thread contexts for vertex processing” and “[t]he same would hold true if enough shared resources were not available for vertex processing, then Fiske would prioritize and process pixel operations,” none of its citations support that, and its expert merely parrots Petitioner, without any analysis or evidence supporting such an interpretation of Selzer. *Id.*

Rather, in Selzer, the rendering module (the alleged unified shader) will only begin to process geometry operations if it has no rendering operations to perform. EX1009, 47. That is, like other load balancing schemes, the only consideration of when to switch is how much work of a given type there is to do. EX2121, ¶371. There is no disclosure in Selzer that it switches from rendering operations to geometry operations on the basis of having enough available space for geometry operations in any of Petitioner’s identified memories. *See id.* Fiske likewise contains no disclosures that available space is checked prior to executing certain threads; what thread it executes is purely based on priority. EX2010, 211.

Further, the claims have specific differences beyond the evaluation of storage space that Selzer and Fiske fail to disclose. EX2121, ¶372. Claim 1 requires performing vertex operations until there is no more room in the general purpose register block for more incoming vertex data, in which case continuing to perform pixel operations until enough space appears to once again store vertex data. *Id.* But Selzer’s rendering module, the alleged unified shader, doesn’t do that. Instead, it operates on rendering data, which Petitioner alleges to be the claimed pixel data (Pet., 68), and then when it runs out of the alleged *pixel* data, it operates on geometry data, which Petitioner alleges to be vertex data (*id.*). There is no disclosure at all when to cease sending, storing, or operating on the alleged *vertex* data, much less any description of doing so based on not having room for vertex data in the general purpose register block, nor any description of continuing to execute pixel processing in that scenario. EX2121, ¶372. There is also no description of determining a second time at which there is again room to store vertex data. *Id.*

Claim 3 requires the processor unit to be “operative to perform pixel calculation operations until enough shared resources become available and then use the shared resources to perform vertex calculation operations.” EX2121, ¶373. Similar to the above, the triggering event in Selzer in going from the alleged pixel calculations (rendering data) to the alleged vertex calculations (geometry data), has

nothing to do with a determination that at a given time, there is finally enough shared resources for geometry data; the only consideration is that there is no more rendering data to operate on. *Id.* There may have been sufficient resources before that happens, but the system has no way of knowing, and thus no way of “perform[ing] pixel calculation operations *until* enough shared resources become available and then us[ing] the shared resources to perform vertex calculation operations.” *Id.*

Claim 4 is like claim 3 but in the alternative direction; it requires the processor unit to be “operative to perform vertex calculation operations until enough shared resources become available and then use the shared resources to perform pixel calculation operations.” EX2121, ¶374. There is no disclosure in Selzer of processing the alleged vertex data (geometry data) until there is enough room for processing the alleged pixel data (rendering data). *Id.* There is not even disclosure of a switch in that direction. *Id.* The only switch described for the rendering module is switching from rendering data to geometry data, not from geometry data to rendering data. *Id.* And again, even the switch in the wrong direction is only done based on whether or not there is any rendering work to do. *Id.*

2. No Processor Unit That Performs Vertex Manipulation Operations And Pixel Manipulation Operations At Various Degrees Of Completion Based On Switching Between Instructions In The Instruction Store (Claim 11)

This claim requires the processor unit to switch between unfinished vertex operations and pixel operations. EX2121, ¶375; EX2123, 61:16-62:2. Petitioner relies on purely on Fiske, which doesn't disclose this functionality. Pet., 78.

Petitioner relies on Fiske's thread priority system. Pet., 72 ("Because Fiske assigns a priority to each and every thread in the system, the scheduling and instruction selection may be very fine-grained and lead to an efficient and dynamic system that is able to move back and forth between vertex and pixel operations very rapidly."), 73-74 (describing Fiske's scheduling algorithm). "[T]he goal is to identify as exactly as possible a relative order in which threads should be run, and devote as many resources as possible to the most important threads." *Id.* Fiske describes loading the highest priority threads to the processor. *Id.* Loaded threads are then scheduled by Fiske's scheduler for execution by the processor. *Id.*

This doesn't disclose switching mid-thread, but even if it did, Petitioner's expert's theory is that the *instructions* that are constituent parts of the thread are the "operations" of claim 11. EX2123, 86:11-15 ("Fiske describes switching among different instructions, so the instructions are the operations" of claim 11); EX2121, ¶¶375-376. And Fiske says nothing about switching between incomplete threads, much less incomplete individual instructions, and Petitioner hasn't

identified anything. *Id.* Rather, Petitioner’s arguments are entirely conclusory, and not tied to any specific disclosure in Fiske. For example, Petitioner claims “Fiske performs another selection to determine which resources should be used for vertex or pixel operations,” when in fact Fiske doesn’t even mention vertex or pixel operations. Pet., 74; EX1010; EX2121, ¶376.

However, to the extent Fiske does disclose switching execution between unfinished threads, and further unfinished instructions, it doesn’t do so between *vertex and pixel* threads and instructions because it is indifferent to vertex and pixel operations; any purported switching occurs only between loaded high priority threads. EX2121, ¶377; EX1010, 211. Fiske doesn’t discuss graphics processing at all. *Id.*

Moreover, even if Fiske disclosed what Petitioner alleges, there is no motivation to add that functionality to Selzer. EX2121, ¶378. Selzer wants its rendering module to render if there is any rendering work to do, because that results in “peak performance.” *See* EX1009, 47-48. That is, only if it is in an “idle state,” which occurs “if there is no rendering data in his input buffer,” does the rendering module “request new unprocessed geometry objects and continue geometry calculations.” *Id.* For the rendering module to switch between rendering and geometry, it “consumes time.” *Id.*, 48. Therefore, a POSITA would not add the ability to switch between unfinished pixel and vertex threads to Selzer, which

would decrease Selzer’s efficiency at processing graphics in the manner it deems most efficient and with highest performance, with no upside. EX2121, ¶378.

Separately, Petitioner hasn’t explained how Fiske (or Selzer) would switch between incomplete pixel and vertex operations “*based on* switching between instructions in the instruction store.” EX1001, Claim 11 (emphasis added). Petitioner merely asserts that switching between instructions of different threads meets the limitations of claim 11 but doesn’t describe how those instructions are located within an instruction store, which Petitioner alleges is Selzer’s master module. Pet., 74; EX2121, ¶379.

3. No Execution of Instructions “In Response to” Receiving Selected Data (Claim 2)

Petitioner doesn’t even allege that the execution by Selzer’s alleged processor unit is done in response to selected data. Pet., 70-71 (regarding the “processor” element); Pet., 66-72 (regarding claim 2 of the ’454 Patent); EX2121, ¶380. Without such an allegation, the analysis can end there, and the ground fails as to claim 2.

Fiske itself states the opposite. EX2121, ¶381. The ground relies on Fiske’s “thread prioritization scheduling scheme,” which is “a simple means of guiding the scheduling of threads on a node,” to explain how the alleged combination decides what the processor will execute and when. Pet., 70-72; EX1010, 211. Fiske confirms that it is the receipt and loading of threads, or groups of instructions, that

cause execution—not receipt of data. “The software scheduler uses the priority to decide *which threads are loaded* . . . the hardware scheduler uses the priority to determine *which loaded thread to execute*.” EX1010, 211; EX2121, ¶341; EX2121, ¶381.

The ID held that “performing rendering or geometry processing based on the data received” meets the claim requirement. First, there was no allegation in the petition regarding this theory, and it is improper for the Board to add new theories. *In re Magnum Oil Tools Int’l, Ltd.*, 829 F.3d 1364, 1381 (Fed. Cir. 2016). Second, it is incorrect. EX2121, ¶382. The fact that somehow operations are done on data doesn’t mean that that the “processor unit executes instructions . . . in response to selected data.” *Id.* All Selzer discloses is that the rendering module receives data, and at some point processes it. EX1009, 41. If the triggering event is the receipt of instructions, as it is in Selzer and Fiske, rather than receipt of data, the execution simply is not “in response to selected data,” it is simply *on* selected data, in response to receiving instructions. EX2121, ¶382.

4. Alleged “Sequencer,” “Instruction Store,” “Circuitry,” “Selection Circuit,” and “Arbiter” Are Not Within Alleged “Unified Shader” As Required (Claims 2, 5-7, 10, 11)

Claims 2, 5, and 11 each require a “sequencer” that “maintain[s] instructions” or “instruction store” that stores in instructions, each of which being

part of the unified shader. *See* Section V.B.4 (providing preamble is limiting), *supra*; EX2121, ¶383.

But the alleged sequencer and the alleged instruction store in Ground 3 are not within the alleged unified shader, and thus the ground fails as to claims 2, 5, and 11. EX2121, ¶384. Petitioner identifies Selzer’s “rendering module” as the claimed unified shader. Pet., 67, 75. Petitioner identifies Selzer’s master module as the claimed sequencer, as its expert confirmed. Pet., 71 (in section regarding “sequencer” of claim 2, describing only abilities of master module); EX2123, 101:4-15.

But the master module is outside the rendering module, not inside it as required. EX2121, ¶385. Selzer describes the master module and the rendering module as “different units.” EX 1009, 40. This is also seen in Figures 4.2 and 4.4 of Selzer:

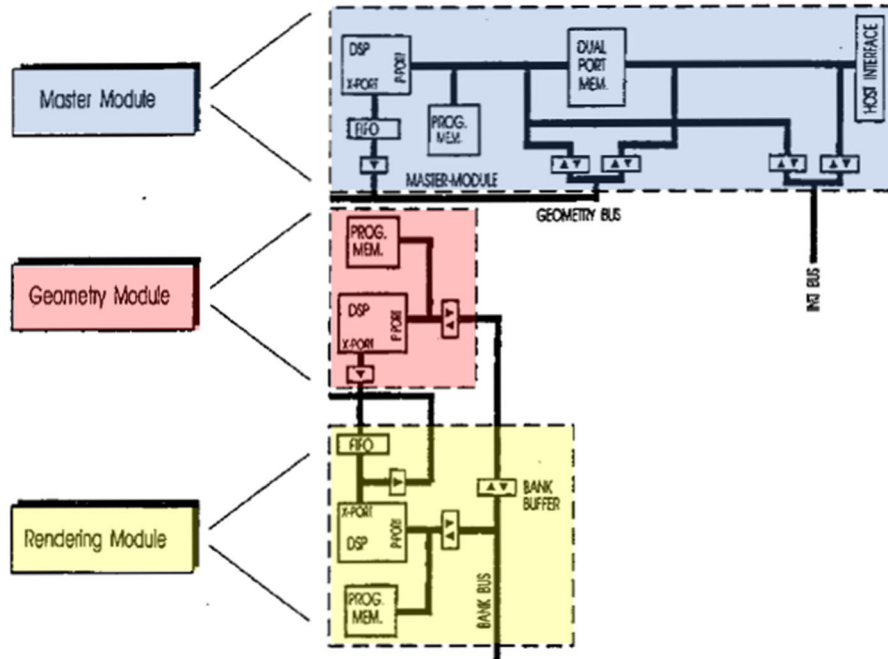


FIGURE 4.2. Basic module structure

EX1009, 42 (annotated).

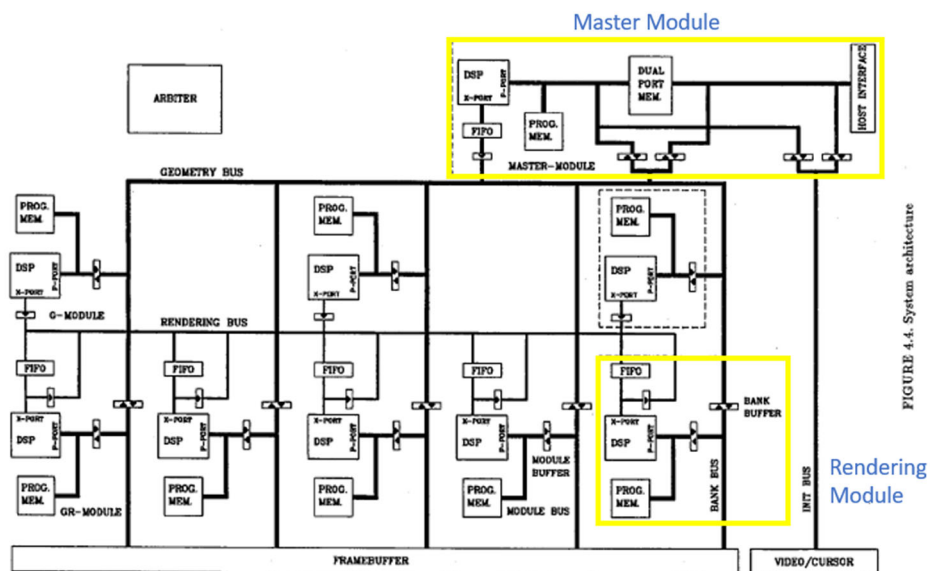


FIGURE 4.4. System architecture

EX1009, 44 (annotated).

Petitioner identifies Selzer's master module as also meeting the "circuitry" of claim 6. Pet., 75. Thus the alleged "circuitry" of claim 6 within the sequencer of claim 5 is not within the alleged unified shader, either. EX2121, ¶386.

Petitioner's theories with respect to "selection circuit" and "arbiter" also fail. EX2121, ¶387. Petitioner doesn't clearly identify what it alleges to be the "selection circuit," it only says it is "shown below" with an excerpt of Selzer's Figure 4.4 with no annotations to which components it is identifying. *Id.*; Pet., 76. To the extent that it is the master module and/or the "Arbiter" of Figure 4.4, these components are also not within the claimed unified shader as required in claim 7 and 10. EX2121, ¶387. To the extent the "arbiter" is the DSP of the master module, this also fails. *Id.*

5. Alleged "Sequencer" And "Instruction Store" Do Not Maintain Instructions (Claims 2, 5, and 11)

Separately from the failures described in Section V.C.4 above, the master module (alleged sequencer and instruction store) of Selzer doesn't satisfy the requirement of claims 2, 5, and 11 of "maintaining instructions" or storing "instructions in the instruction store." Ex. 2121, ¶388; EX1001, claims 2, 5, and 11.

Petitioner's theory is that "the master module in Selzer accesses the graphics data in the system and 'fetches the next primitive or task automatically.'" Pet., 71. But as Petitioner's expert agreed in deposition, the Petitioner and his declaration

were incorrect. EX2123, 107:6-23. It is the geometry module—not the master module—that performs the function that Petitioner relies on as its evidence of why this limitation is disclosed. EX1009, 43 (“If a geometry module has finished the last task, it accesses the geometry bus and fetches the next primitiv[e] or task automatically.”); EX2123, 107:6-17 (agreeing that “it’s the geometry module” that fetches the next primitive or task automatically and his report was erroneous); EX2121, ¶389. The geometry module wholly distinct from and not a part of the master module, as described above, and shown in Figure 4.2 of Selzer. *Id.*

Petitioner argues that Selzer also says the master module is the “systems supervisor, handles communication to the host processor and is responsible for start-up and synchronizing activities.” Pet., 71. Nothing here states the master module “maintains instructions,” as required. EX2121, ¶390; EX1009, 40. The master module’s role is “put[ing] the high order primitives like splines, polygons, meshes or triangles,” *i.e.*, data, “into the data FIFO.” *Id.* As Petitioner’s expert conceded, the claimed instructions are different from mere data. EX2123, 24:14-25:1.

Petitioner further argues that “[t]he master module gathers the graphics data and tasks and supplies those tasks (data and instructions) to the appropriate processor unit.” Pet., 71-72. But that is incorrect. EX2121, ¶391. All Selzer says is that the master module puts data into the data FIFO. EX1009, 41 (defining “The

Master Modul [*sic*]” by stating: “The master module traverses the graphics data structure and feeds graphics data to a special first-in-first-out memory (FIFO) for delivering to the appropriate processors.”); 43 (“The master module traverses the graphics data structure and puts the high order primitives like splines, polygons, meshes or triangles into the data FIFO.”). As explained above, it is the geometry module that handles instructions. EX2121, ¶391.

Petitioner further argues that Fiske in combination with Selzer also discloses the sequencer. Pet., 72. This theory builds on Selzer according to Petitioner. *Id.* But Petitioner doesn’t actually identify any component in Fiske as the claimed sequencer, and focuses on adding Fiske to Selzer’s rendering module, not the alleged sequencer, the master module. *Id.* The alleged ability of Fiske to assign priorities on a per thread basis doesn’t disclose a sequencer, nor does Petitioner say so. EX2121, ¶392.

6. No “Arbiter” (Claim 10) or “Control Signal” (Claims 7, 10)

Petitioner seemingly identifies the DSP of the master module as meeting the claimed “arbiter” required in claim 10. Pet., 76-77; EX2121, ¶393. However, Selzer doesn’t disclose any arbitration abilities by the DSP, nor the provision of a control signal. EX1009, 39-40. Thus, Petitioner’s theory is unsupported by Selzer’s disclosures.

Additionally, Petitioner failed to identify any “control signal” outside of conclusory allegations. EX2121, ¶394. The petition alleges that the DSP of the master module generates a control signal but cites to nothing from Selzer. Pet., 76. Dr. Pfister cites to Selzer pages 43-45 but they do not disclose any control signal that is allegedly generated by the DSP of the master module, nor do any others. EX1003, ¶251; EX2121, ¶394. Selzer contains only two references to DSP, and none of them describe anything regarding a control signal. *Id.*; EX1009, 39-40.

Petitioner also alleges that the combination of Selzer and Fiske discloses claims 7 and 10. Pet, 76-77. Petitioner identifies the master module of Selzer and the scheduling arbiter of Fiske as the selection circuit, but the same issues described above apply here in that the master module is part of the alleged unified shader, the rendering module. EX2121, ¶395.

With respect to claim 10, Petitioner identifies Fiske’s hardware scheduler as the claimed arbiter and claims that its “simple hardware” provides a control signal. But there is no disclosure of a control signal provided by the “simple hardware” in Fiske. EX2121, ¶396. All Fiske states is: “On a context switch, the hardware scheduler chooses the loaded thread with the highest priority as the next thread using simple hardware, in order to minimize context switch overhead.” EX1010, 211. Nor has Petitioner shown or alleged that the alleged arbitration disclosed by Fiske requires providing a control signal. EX2121, ¶396.

7. No Motivation to Combine Selzer and Fiske

Petitioner's arguments for motivation to combine amount to nothing but conclusory statements. Petitioner asserts that it would be obvious for a POSITA to combine Selzer and Fiske because both are directed to dynamic load balancing schemes. Pet., 64. But Fiske literally says that it is not: "We do not consider dynamic load balancing issues in this paper." EX1010, 211; EX2121, ¶397.

Further, combining Fiske with Selzer because it would be antithetical to Selzer's disclosure that its "peak performance" is when "all rendering processor power is exploited for rendering calculations and the rendering are working continuously." EX1009, 48. It would make no sense to switch away from processing rendering calculations if such calculations were unfinished, as Petitioner suggests is disclosed by Fiske. EX2121, ¶398. Doing so would lose the system's peak performance. *Id.*

The only time it makes sense for Selzer's rendering module to work on geometry is when "there is no rendering data in [the rendering processor]'s input buffer." EX1009, 47. But Selzer already discloses an adequate task switching ability which maintains its peak performance. *Id.* Thus, adding Fiske solely for the purpose of switching threads, as opposed to tasks, is unnecessary and not something a POSITA would do. EX2121, ¶399.

Finally, Selzer provides very little details regarding what DSP it uses.

EX1009, 39-40; EX2121, ¶¶404-405. Fiske's architecture is very different from Selzer as it is focused on highly parallel computing systems where threads interact through communications and synchronization methods. *Id.*, ¶¶400-404. Fiske is a research prototype with innumerable design details that need to be sorted prior to actual implementation requiring a significant amount of engineering design, research, and validation. *Id.*, ¶¶406-407. Given the lack of details provided by Selzer and Fiske, Petitioner hasn't shown that a POSITA would have a reasonable expectation of success in combining them. *Id.*

D. Secondary Considerations

Secondary considerations support non-obviousness of the '454 patent. Further, the '454 patent is entitled to a presumption of nexus because the inventions of the '454 patent developed in the R400 project are embodied by and are coextensive with a specific product – the unified shader used in the Xenos GPU of the highly successful gaming console Xbox360. EX2102 (Lefebvre Dep), 132:1-149:25, 198:1-199:25, 240:1-241:25, 246:6-14; EX2133, 5; EX2145 (Andrews), 13; EX2127 (Luebke), 99; EX2121, ¶¶409-410. Even if not entitled to nexus, the objective indicia are the direct result of the unique characteristics of the inventions claimed in the '454 patent, *e.g.*, the unified shader. *Id.*

1. Initial Skepticism of ATI's Unified Shader

ATI's unified shader was greeted with skepticism. EX2121, ¶411. For example, industry analysts opined that unified shaders were “not the obvious right answer right now,” and that Microsoft was “taking a chance” by undertaking the “experiment” of unified shaders. EX2144 (Takahashi), 5.

Notably, companies such as competitor NVIDIA, publicly criticized the unified shader, characterizing it a “Jack of all trades, master of none.” EX2121, ¶412; EX2129 (Shilov), 1; EX2130 (Shilov), 1 (“It’s far harder to design a unified processor - it has to do, by design, twice as much.”). Sony agreed: “NVIDIA’s approach is superior.” EX2121, ¶413; EX2146 (Kutaragi), 5.

Industry members highlighted the difficulties with unified shader technology. EX2121, ¶414; EX2136 (Shimpi), 7 (“[I]t is even more imperative to make sure that everything is running at maximum efficiency” for unified shaders.). Others expressed uncertainty regarding the success of unified shader technology. *Id.* (“Performance . . . is a very difficult aspect to assess without testing.”); EX2124 (Beyond3D), 32; EX2121, ¶415. The inventors had to even convince others within ATI about the merits. EX2113, 39:23-44:21 (providing “skepticism” if the idea would work and “concern” regarding performance).

2. Unexpected Results in Developing the Unified Shader

ATI's revolutionary efforts in developing its unified shader technology amounted to significant, innovative, and disruptive advances over existing graphics processors. EX2121, ¶416. When details about ATI's unified shader became known through the Xenos GPU, the "fresh perspective" of ATI "force[d]" those in the industry to "reconsider" the graphics pipeline. EX2124, 1-2. ATI engineers themselves also reported experiencing some unexpected results during development. EX2121, ¶417; EX2144, 5; EX2147 (Wasson), 1.

3. The Unified Shader's Satisfaction of a Long-Felt Need and Failed Attempts by Others

The unified shader was a fundamentally different approach than the current graphics technology available at the time of its launch. EX2121, ¶418; EX2124, 3 (the unified shader is "different from others seen before," and has "a fundamental difference with virtually all current shader capable graphics processors."); EX2145, 8; EX2147, 1.

This fundamentally different approach of the unified shader satisfied long-felt needs in the industry "[f]or years," where others failed to satisfy those needs. EX2121, ¶419; EX2136, 6; EX2133 (How Xbox360 Works), 5 ("ATI figured out a way at the hardware level to address the needs of both types of shaders using the same pipeline.").

4. Industry Praise of the Unified Shader in the Xenos GPU

ATI's unified shader technology received praise in the industry by experts and competitors. EX2121, ¶¶420-422; EX2131 (Fortune), 3; EX2134 (Kuchera), 8; EX2136, 7; EX2133, 4; EX2140 (NeutralX2), 1-3 (describing "innovative use" of unified shader architecture as "something special."); EX2127, 99 ("The net result is a flexible parallel architecture that improves GPU utilization and provides much greater flexibility for game designers."); EX2138 (IGN), 5; EX2139 (RedGamingTech), 1.

5. Commercial Success of the Xbox360 Containing ATI's Unified Shader

Despite some skepticism and uncertainty, the new unified shader technology experienced significant commercial success, with the Xenos GPU and the Xbox360 generating billions of dollars in revenue for ATI and for Microsoft. EX2121, ¶¶423-424; EX2132 (Techspot), 13 ("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 Xbox360."); EX2128 (Microsoft), 4; EX2135 (Pensky), 1; EX2142 (Statista), 1 (Xbox360 accounting for 65% of global market share of video game consoles in 2006).

The commercial success of the Xbox360 is attributable to unified shader technology of the '454 patent in its Xenos GPU. EX2121, ¶425. This is evidenced by the importance of the unified shader with respect to the viability of the

Xbox360 in the market and consumer preferences. EX2136, 10-11; EX2133, 4 (“[T]he most innovative thing about th[e Xenos GPU] is that it is built on unified shader architecture.”).

6. Adoption by Others

AMD’s unified shader has been widely adopted by others in the industry, such as Qualcomm and NVIDIA. EX2121, ¶¶426-428; EX2137 (Qualcomm), 2 (“Qualcomm licensed a GPU from ATI, call[ing] it Adreno 200.”); EX2141 (Wikipedia-Adreno), 2 (showing Adreno 200 and later models with “Unified shader”); EX2143 (NVIDIA), 2 (“Historically, GPUs have had dedicated units for . . . vertex processing and pixel shading. With the *unified architecture of the GeForce 8Series*, NVIDIA designed a single floating point shader core with multiple independent processors.”).

VI. CONCLUSION

For any of the foregoing reasons, the Board should determine that Petitioner has failed to prove that the challenged claims of the ’454 Patent are unpatentable.

Dated: February 23, 2024

Respectfully submitted,

/s/ William A. Meunier

William A. Meunier (Reg. No. 41,193)

Michael T. Renaud (Reg. No. 44,299)

Adam S. Rizk (Reg. No. 66,867)

MINTZ, LEVIN, COHN, FERRIS,

GLOVSKY AND POPEO, P.C.

One Financial Center

Boston, MA 02111

Telephone: 617-542-6000

Facsimile: 617-542-2241

WAMeunier@mintz.com;

MTRenaud@mintz.com

ARizk@mintz.com

Reza Dokhanchy (Reg. No. 62,795)

MINTZ, LEVIN, COHN, FERRIS,

GLOVSKY AND POPEO, P.C.

3580 Carmel Mountain Road, Suite 300

San Diego, CA 92130

Telephone: 858-314-1596

RDokhanchy@mintz.com

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24(d), Patent Owner hereby certifies, in reliance on the word count of the word-processing system (Microsoft Office Word 2010) used to prepare this preliminary response, that the number of words in this paper is 13,988. This word count excludes the tables of contents, tables of authorities, certificate of word count, certificate of service, and table of exhibits.

Dated: February 23, 2024

/s/ William A. Meunier

William A. Meunier (Reg. No. 41,193)

CERTIFICATE OF SERVICE

I certify that a copy of Patent Owner's Response and accompanying ATI Exhibits 2121 through 2147 is being served by electronic mail on the following counsel of record:

Lead Counsel

Jeffrey Johnson
Reg. No. 53,078
ORRICK, HERRINGTON &
SUTCLIFFE, LLP
609 Main Street, 40th Floor
Houston, TX 77002-3106
Telephone: (713) 658-6400
Facsimile: (713) 658-6401
Email: 3J6PTABDocket@orrick.com;
Realtek-AMD_OHS@orrick.com

Backup Counsel

Christopher J. Higgins
Reg. No. 66,422
ORRICK, HERRINGTON &
SUTCLIFFE, LLP
1152 15th Street, N.W.
Washington, DC 20005-1706
Telephone: (202) 339-8400
Facsimile: (202) 339-8500
Email: OCHPTABDocket@orrick.com;
Realtek-AMD_OHS@orrick.com

Steve Baik
Reg. No. 42,281
WHITE HAT LEGAL
P. O. Box 382
San Jose, CA 95002
Telephone: (650) 618-5282
Email: sbaik@whitehat.legal

Dated: February 23, 2024

/s/ William A. Meunier

William A. Meunier (Reg. No. 41,193)