

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,

v.

ATI TECHNOLOGIES ULC,
Patent Owner.

IPR2023-00922
Patent 8,760,454 B2

Before JAMES P. CALVE, BRIAN J. McNAMARA, and
KEVIN W. CHERRY, *Administrative Patent Judges*.

McNAMARA, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

Realtek Semiconductor Corp. (“Petitioner”) filed a petition, Paper 1 (“Petition” or “Pet.”), to institute an *inter partes* review (“IPR”) of claims 1–11 (the “challenged claims”) of U.S. Patent No. 8,760,454 B2 (“the ’454 patent”). 35 U.S.C. § 311. ATI Technologies ULC (“Patent Owner”) filed a Preliminary Response, Paper 6 (“Prelim. Resp.”), contending that the Petition should be denied as to all challenged claims. We authorized Petitioner’s filing of a Preliminary Reply to Patent Owner’s Preliminary Response. (Paper 8, “Prelim. Reply”) and Patent Owner’s filing of a Preliminary Sur-reply (Paper 9, “Prelim. Sur-reply”).

We have jurisdiction under 35 U.S.C. § 6. This Decision on Institution is issued pursuant to 35 U.S.C. § 314, which provides that an *inter partes* review may not be instituted unless the information presented in the Petition “shows that there is a reasonable likelihood that the Petitioner would prevail with respect to at least 1 of the claims challenged in the petition.”

A decision to institute under § 314 may not institute on fewer than all claims challenged in the petition. *SAS Inst., Inc. v. Iancu*, 138 S. Ct. 1348, 1359–60 (2018). In addition, per Board practice, if the Board institutes trial, it will institute “on all of the challenged claims and on all grounds of unpatentability asserted for each claim.” *See* 37 C.F.R. § 42.108(a).

Having considered the arguments and the associated evidence presented in the Petition and the Preliminary Response., for the reasons described below, we institute *inter partes* review.

II. REAL PARTIES IN INTEREST

The Petition identifies Realtek Semiconductor Corp. (“Realtek”) as its real party-in-interest. Pet. 1. Patent Owner identifies ATI Technologies ULC as its real party-in-interest. Paper 4, 1.

III. RELATED MATTERS

Petitioner and Patent Owner identify the following proceedings as ones that may affect or be affected by a decision in this proceeding:

Advanced Micro Devices, Inc. et al v. TCL Industries Holdings Co., Ltd. et al, C.A. No. 2:22-cv-00134 (E.D. Tex. May 5, 2022) (“the AMD Litigation”); and

Certain Graphics Systems, Components Thereof, and Digital Televisions Containing the Same, Inv. No. 337-TA-1318 (“the ITC Investigation”).

Pet. 2, Paper 4, 1. Petitioner states that the AMD litigation is stayed pending completion of the ITC Investigation, expected on or about November 7, 2023. Pet. 2.

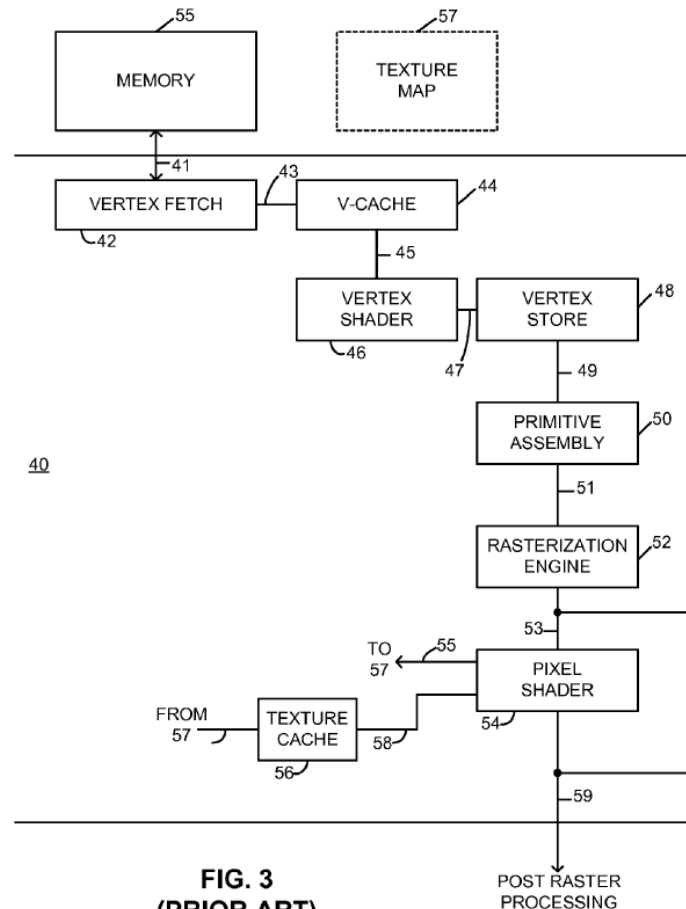
IV. THE ’454 PATENT

The ’454 patent concerns a graphics processing architecture that employs a single or “unified shader,” i.e., a shader that “is configured to perform both vertex and pixel operations.” Ex. 1001, 1:32–33, 3:10–12. The ’454 patent explains that, in computer graphics, complex shapes and structures are formed by sampling, interconnecting, and rendering simpler objects, e.g., triangles or other suitable polygons, called primitives. *Id.* at 1:38–42. Primitives are formed by interconnecting individual pixels. *Id.* at 1:42–43. In order to render an object for display, based on the location of the pixels within the primitives and the primitives’ orientation with respect

to the desired shape, color and texture are applied to the individual pixels that make up the shape to be generated. *Id.* at 1:42–48.

Graphics processors that interconnect the primitives and apply color and textures to the generated shapes include a series of shaders that specify how a final image is drawn on a display device and its corresponding attributes. *Id.* at 1:49–54. A shader receives shape data in object space (x , y , z), color information, texture information, luminance information, and viewing angle information and produces output data (x' , y' , z') that represent the object with texture and other appearance properties applied to it. *Id.* at 1:55–60. Figs. 2A, 2B of the '454 patent show vertex data V_x , V_y , V_z of a cube applied to a vertex shader that outputs angularly oriented vertices V_x' , V_y' , V_z' and appearance attributes of a corresponding cube. *Id.* at 2:3–7. A pixel shader operating at the pixel level provides the color value associated with each pixel of a rendered object. *Id.* at 2:8–12.

According to the '454 patent, in a conventional graphics processor, the vertex shader and pixel shader are “separate components that are configured to perform only a single transformation or operation. Ex. 1001, 2:12–15. “In conventional graphics processors, the vertex shader and the pixel shader are juxtaposed in a sequential, pipelined fashion, with the vertex shader being positioned before and operating on vertex data before the pixel shader can operate on individual pixel data.” *Id.* at 2:25–29, 4:5–7. Figure 3, reproduced below, is a schematic of such a conventional shader.



In graphics processor 40 of Figure 3, on line 41 vertex fetch block 42 receives from off-chip memory 55 vertex information relating to primitives to be rendered. *Id.* at 3:23–27. The fetched vertex data (e.g., object shape, color and texture information, viewing angle) is stored in vertex cache (V-cache) 44 and, upon request, is transmitted on line 45 to vertex shader 46. *Id.* at 3:26–30. Vertex shader 46 typically is programmed to apply a transformation position matrix to the input position information supplied from V-cache 44 and thereby provide data representing a perspective corrected image of the object to be rendered, as well as texture and color coordinates for the image. *Id.* at 3:30–39. Vertex shader 46 transmits the transformed vertices to vertex store 48, which then transmits the modified

vertex information to primitive assembly block 50, where the input vertex information is converted to primitives using well known techniques. *Id.* at 3:40–49. Rasterization engine 52 receives and converts the previously assembled primitives into pixel data that is then transmitted to pixel shader 54. *Id.* at 3:49–53. Pixel shader 54 generates color and additional appearance attributes for each pixel and applies the attributes to respective pixels. Pixel shader 54 can fetch texture data from texture map 57, as indexed by pixel data from rasterization engine 52, and apply logical or arithmetic operations on the received texture data to generate the pixel color or other appearance attributes of interest. *Id.* at 3:54–66. The pixel appearance attribute is also combined with a base color provided by rasterization engine 54 to provide a pixel color corresponding to the position of interest. *Id.* at 3:66–4:4.

To avoid the computational inefficiency and space requirements of separate vertex and pixel shaders, the '454 patent incorporates into the graphics processor a unified shader that performs both pixel and vertex operations, as shown in Figures 4A and 4B. Ex. 1001, 2:45–46, 4:5–28. Figure 4A, reproduced below, is an exemplary embodiment of a graphics processor with a unified shader. *Id.* at 3:47–48, 4:13.

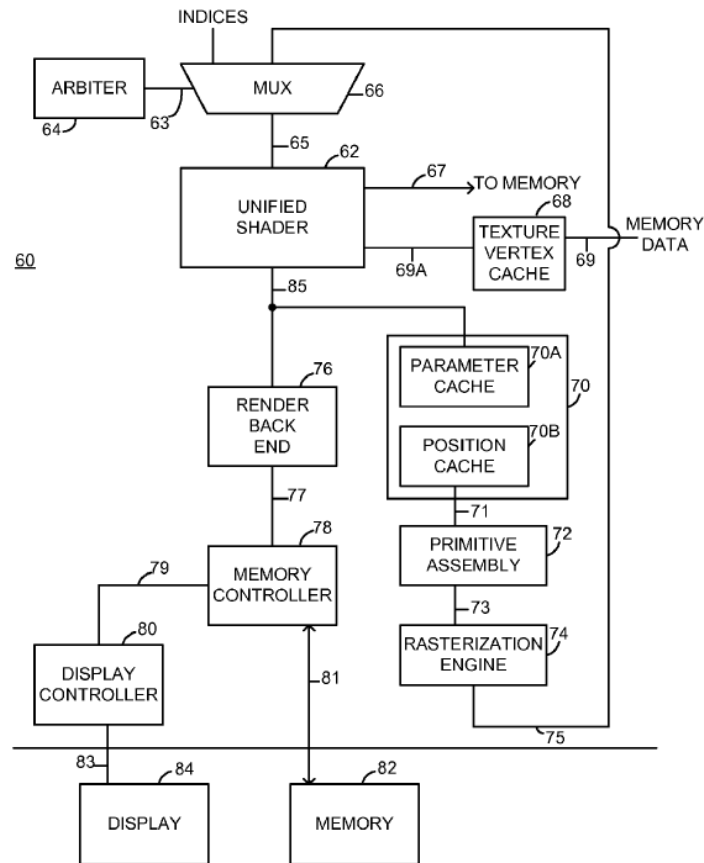
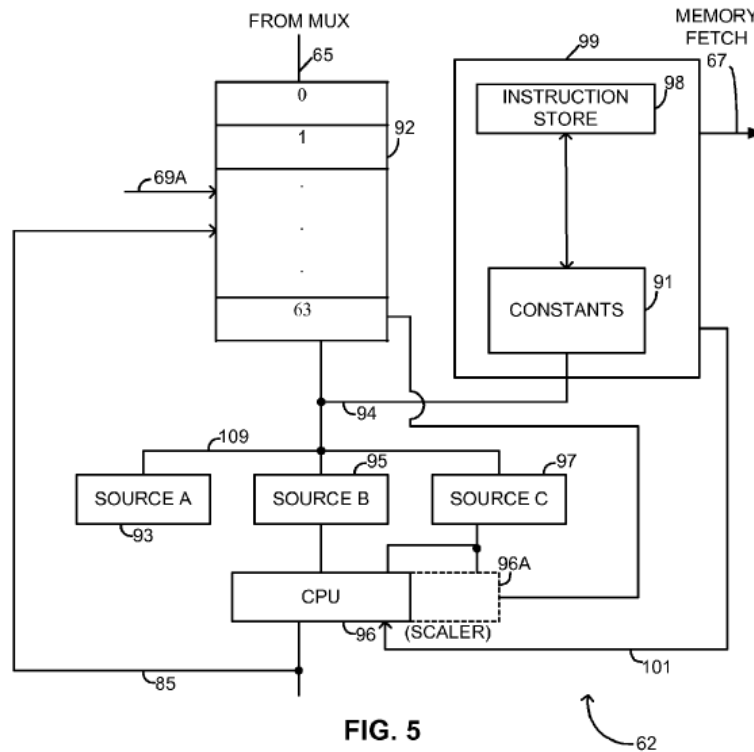


FIG. 4A

Ex. 1001, Fig. 4A. As Figure 4A illustrates, graphics processor 60 includes arbiter 64, multiplexer (mux) 66, and unified shader 62. A control signal on line 63 from arbiter 64 to mux 66 determines which of two inputs mux 66 supplies to unified shader 62. *Is.* At 4:19–21. Mux 66 sends unified shader 62 vertex data (e.g., indices) from a first mux input if there are enough resources available in the shader to operate on the vertex data; otherwise, mux 66 sends unified shader 62 interpolated pixel parameter data from rasterization engine 74 on line 75. *Id.* at 4:13–28.

Figure 5, reproduced below, is a schematic of unified shader 62.
 Ex. 1001, 4:31–32.



Id. at Fig. 5. Shader 62 includes (i) register block 92 (comprising 64 general purpose registers), (ii) source registers A, B, and C (reference designators 94, 95, and 97, respectively), (iii) CPU 96 (adapted to perform 32-bit floating point arithmetic operations and logical operations on corresponding operands and, in section 96A scalar operations, e.g., log, exponent), and (iv) sequencer 99. *Id.* at 4:40–52. Sequencer 99, which includes instruction store 98, determines whether the next instruction to be executed from instruction store 98 on data maintained in general purpose register 92, as provided by the source registers, is an arithmetic instruction, or a logical instruction or a memory instruction, e.g., a texture fetch. *Id.* at 4:52–66. Sequencer 99 also includes constants block 91—constants block 91 includes transformation matrices used in connection with vertex manipulation operations. *Id.* at 4:53–55. Instruction store 98 maintains both vertex manipulation and pixel manipulation instructions, enabling the shader to

perform both vertex and pixel operations, and to execute memory fetch operations based on the information received from mux 66. *Id.* at 5:19–27.

In application, when vertex data (e.g., indices) to be processed is transmitted to general purpose register block 92 from mux 66, instruction store 98 passes corresponding control signals on line 101 to processor 96 to perform vertex operations. If general purpose block 92 does not have enough space available to store the incoming vertex data, arbiter 64 will not instruct mux 66 to transmit vertex data. *Id.* at 5:31–44. Instead, any pixel calculations currently being performed by processor 96 are continued based on instructions from instruction store 98, until enough registers in block 92 become available to accommodate the vertex data. *Id.* at 5:45–49. Unified shader 62 can switch from executing vertex to pixel instructions, regardless of the degree of completion, reducing the down time of processor 96. *See id.* at 5:32–52.

Pixel based output from the unified shader on line 85 is sent to cache block 70 that includes parameter cache 70A and position cache 70B. Ex. 1001, 5:54–58, Fig. 4A. Pixel information in cache block 70 is transmitted to primitive assembly block 72, where the information is assembled into a series of triangles or other primitives and transmitted to rasterization engine block 74. Rasterization block 74 converts the primitives to individual pixel data, e.g., through a walking process, and interpolated pixel data is transmitted to the second input of the mux on line 75. *Id.* at 5:63–6:4.

Vertex data output from shader 62 on line 85 is transmitted to back end block 76, where the vertex data is converted to a format suitable for display on display device 84, and transmitted to memory 82 and display controller 80, where the formatted information is routed to display 84. Ex. 1001, 6:5–18.

V. ILLUSTRATIVE CLAIMS

Independent claim 1 is drawn to a method carried out by a unified shader. Ex. 1001, 6:52–7:11. Independent claims 2, 3, 4, 5, and 11 are drawn to a “unified shader,” a term in the preambles that limits the claims to shaders that can perform both vertex and pixel processing. Ex. 1001, 7:13–8:30. *See* Section VIII.B herein. Claims 6–9 depend from claim 5. Claim 10 depends from claim 7.

Independent claim 2, reproduced below using Petitioner’s paragraph designations, is illustrative:

- 2. [pre] A unified shader, comprising:
 - [a] a general purpose register block for maintaining data;
 - [b] a processor unit;
 - [c] a sequencer, coupled to the general purpose register block and the processor unit, the sequencer maintaining instructions operative to cause the processor unit to execute vertex calculation and pixel calculation operations on selected data maintained in the general purpose register block; and
 - [d] wherein the processor unit executes instructions that generate a pixel color in response to selected data from the general purpose register block and generates vertex position and appearance data in response to selected data from the general purpose register block.

Ex. 1001, 6:65–7:11. Independent claim 2 recites that the processor unit executes instructions in response to selected data, but does not include a limitation reciting the selection of operations based on the amount of space available in a data store. Independent claim 5, reproduced below, is illustrative of a claim with such an express limitation.

- 5. A unified shader comprising:
 - a processor unit;
 - a sequencer coupled to the processor unit, the sequencer maintaining instructions operative to cause the processor unit to execute vertex calculation and pixel calculation operations

on selected data maintained in a store depending upon an amount of space available in the store.

Ex. 1001, 8:1–8; *see also* claims 1, 3, 4.

Patent Owner characterizes the '454 patent as having several salient features that are embodied in the claims. Prelim. Resp. 9–17.

First, Patent Owner asserts that all of claims 1–11 require a unified shader, i.e., a shader the Specification defines as one that is capable of executing both vertex and pixel threads to transform both vertex and pixel data, respectively¹. *Id.* at 9–11 (citing Ex. 1001, 2:58–61; Ex. 2001, Mangione-Smith Decl. ¶29), 15. The embodiment of the unified shader in the '454 patent includes a register that stores vertex and pixel data that are to be transformed by the unified shader, a sequencer that stores individual instructions that form vertex and pixel threads that the unified shader uses to manipulate and transform the store vertex and pixel data, and a processor that can execute the threads. *See id.* at 10–11.

Second, Patent Owner asserts that the unified shader in the '454 patent allocates the unified shader's resources to switch between unfinished threads to prevent downtime and data backlogs. *See id.* at 11–13; Ex. 1001, 8:22–30 (claim 11).

Third, Patent Owner states that, unlike prior art systems that use pre-determined priorities or balance workloads based on how many of a particular task is in the queue to execute, the '454 patent selects specific data based on the availability of space in the data storage for particular types of data. Prelim. Resp. 13, 15–16. For example, Patent Owner notes that if there is insufficient room in the data storage for vertex data, the processor

¹ We more fully address the construction of the term “unified shader” in Section VIII.B herein.

continues pixel operations until enough registers within the general purpose register block become available to store vertex data. *Id.* at 13–14 (citing, e.g., Ex. 1001, code (57)) 15 (asserting that claims 1 and 3–11 require performing operations based on this dynamically-monitored data storage capacity).

Fourth, Patent Owner asserts that, unlike the prior art, in which a shader performs tasks based on received instructions, the unified shader of the '454 patent is data-driven, such that the receipt of particular selected data dictates the execution of instructions, e.g., the unified shader performs one of the vertex operations or pixel operations based on the selected one of the plurality of inputs. *Id.* at 14 (citing Ex. 1001, 2:58–3:17; Ex. 2001, Mangione-Smith Decl. ¶ 39), 17 (noting that claim 2 recites executing instructions to generate a pixel color and vertex instructions that generate vertex and appearance data in response to selected data).

VI. ASSERTED GROUNDS

Petitioner asserts that claims 1–11 would have been unpatentable on the following grounds:

Claim(s) Challenged	35 U.S.C. §	Reference(s)
1–11	103	Lindholm '685 ² , Lindholm '913 ³

² U.S. Patent No. 7,038,685 to Lindholm et al. issued May 2, 2006 (Ex. 1005).

³ U.S. Patent No. 7,015,913 to Lindholm et al. issued Mar. 21, 2006 (Ex. 1006).

Claim(s) Challenged	35 U.S.C. §	Reference(s)
1–11	103	Amanatides ⁴ , Kohn ⁵
1–11	103	Selzer ⁶ , Fiske ⁷

VII. LEVEL OF ORDINARY SKILL IN THE ART

Petitioner describes a person of ordinary skill (“ordinarily skilled artisan” or “POSITA”) as having “at least a four-year degree in electrical engineering, computer engineering, computer science, or a related field and two years relevant experience in the graphics processing field including developing, designing, or programming hardware for graphics processing units.” Pet. 8. Patent Owner does not comment on the level of ordinary skill.

The level of ordinary skill in the art usually is evidenced by the references themselves. *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001); *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995); *In re Oelrich*, 579 F.2d 86, 91 (CCPA 1978). As Petitioner’s description of a person of ordinary skill is consistent the subject matter of the challenged

⁴ John Amanatides and Edward Szurkowski, *A Simple, Flexible, Parallel Graphics Architecture*, Proceedings of Graphics Interface at 155–160 (Canadian Information Processing Society 1993) (Ex. 1007).

⁵ Les Kohn and Neal Margulis, *Introducing the Intel i860 64-bit Microprocessor*, IEEE, Volume 9, Issue 4, pages 15–30, August 1989 (Ex. 1008).

⁶ Harald Selzer, *Dynamic Load Balancing within a High Performance Graphics System*, In Proceedings of Rendering, Visualization and Rasterization Hardware (Eurographics'91 Workshop) at 37–53 (Springer-Verlag 1993) published in 1993 (“Selzer”) (Ex. 1009).

⁷ Stuart Fiske and William J. Dally, *Thread prioritization: A Thread Scheduling Mechanism for Multiple-Context Parallel Processors*, Proceedings of First Symposium on High-Performance Computer Architecture, 1995 at 210–221 (IEEE 1995) (Ex. 1010).

claims and the references, we apply Petitioner’s description for purposes of this Decision.

VIII. CLAIM CONSTRUCTION

A. *Claim Construction Principles*

For petitions filed after November 13, 2018, we interpret claim terms using “the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. 282(b).” 37 C.F.R. § 42.100(b) (2019). In this context, claim terms “are generally given their ordinary and customary meaning” as understood by a person of ordinary skill in the art in question at the time of the invention. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–13 (Fed. Cir. 2005) (citations omitted) (en banc). “In determining the meaning of the disputed claim limitation, we look principally to the intrinsic evidence of record, examining the claim language itself, the written description, and the prosecution history, if in evidence.” *DePuy Spine, Inc. v. Medtronic Sofamor Danek, Inc.*, 469 F.3d 1005, 1014 (Fed. Cir. 2006) (citing *Phillips*, 415 F.3d at 1312–17). Extrinsic evidence is “less significant than the intrinsic record in determining ‘the legally operative meaning of claim language.’” *Phillips*, 415 F.3d at 1317 (citations omitted).

Any special definition for a claim term must be set forth in the specification with reasonable clarity, deliberateness, and precision. *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994).

We construe only those claim terms that require analysis to determine whether to institute inter partes review. *See Realtime Data, LLC v. Iancu*, 912 F.3d 1368, 1375 (Fed. Cir. 2019) (“The Board is required to construe ‘only those terms . . . that are in controversy, and only to the extent necessary to resolve the controversy.’” (quoting *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999))).

Neither party has proposed a special definition or explicit construction of any claim term. Underlying Patent Owner’s arguments, however, is a construction of the term “unified shader.” Therefore, we consider how “unified shader” should be construed.

B. Unified Shader

All the claims of the ’454 patent recite a unified shader. The ’454 patent states that a unified shader is one “that is capable of performing both the vertex and the pixel operations.” Ex. 1001, 2:45–46. Patent Owner contends that, in addition, a unified shader, as claimed in the ’454 patent, is one that can switch back and forth between executing *unfinished* vertex and pixel threads. Prelim. Resp. 44 (emphasis added) (citing Ex. 2001, Mangione-Smith Decl. ¶ 319); *see also id.* at 11–13 (citing Ex. 1001, 5:23–52, Ex. 2001, Mangione-Smith Decl. ¶¶ 33–35).

Petitioner agrees that “the ’454 patent uses the same piece of hardware, which it also terms a ‘unified shader’ to run both vertex shader programs and pixel shader programs.” Pet. 13 (citing Ex. 1001, 2:58–3:3, 3:10–12). Referring to an embodiment in which processor 96 has one logical partition that performs 32-bit floating point arithmetic operations for vertex processing and another logical partition for processing scalar pixel operations, Petitioner states “[t]he processor unit within the unified shader can process vertex data concurrently with pixel data and alternate between the two types of operations.” *Id.* at 14 (citing Ex. 1001, 5:32–36, 6:36–41; Ex. 1003, Pfister Decl. ¶¶ 54–55). Noting that the unified shader in the ’454 patent maintains instructions for performing vertex and pixel operations in a sequencer, Petitioner acknowledges that an arbiter circuit might give priority to selecting vertex or pixel inputs for processing based on whether the

general purpose registers have space available to store incoming vertex data. *Id.* (citing Ex. 1001, 4:12–28, 4:52–5:5, 5:41–52).

According to Petitioner the '454 patent does not disclose a multithreaded shader, but “at best discloses a pipelined processor that can execute multiple instructions concurrently with respect to different data types, using separate partitions for floating point calculations needed for vertex operations and for scalar/integer pixel operations.” Pet. 41. We understand Petitioner to be arguing that the unified shader of the '454 patent cannot be construed to be one that switches back and forth between unfinished vertex and pixel threads, as Patent Owner argues throughout the Preliminary Response. Although Patent Owner contends that the unified shader of the '454 patent switches between unfinished vertex and pixel threads, the '454 patent does not use the term “thread” and does not refer to or discuss instructions to be executed as a “thread.”

The '454 patent explicitly defines its unified shader stating “[t]he shader of the present invention is referred to as a ‘unified’ shader because it is configured to perform both vertex and pixel operations.” Ex. 1001, 3:10–12. This definition does not limit the unified shader to one that switches back and forth between executing vertex and pixel threads. The '454 patent also describes the embodiment in Figure 4A and Figure 5 as “exemplary.” In this “exemplary” embodiment, the unified shader “has ability to simultaneously perform vertex manipulation operations and pixel manipulation operations at various degrees of completion by being able to freely switch between such programs or instructions, maintained in the instruction store, very quickly.” *Id.* at 5:32–37. Claim 11 explicitly recites this feature. *Id.* at 8:26–30 (reciting “wherein 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”). Based on the explicit definition of a unified shader in the ’454 patent and the explicit recitation in claim 11, we decline to adopt a construction that imports additional limitations into the term “unified shader.”

The limiting term “unified shader” appears in the preamble of all the claims. We construe the claimed “unified shader” to mean a shader that is configured to perform both vertex and pixel operations. Ex. 1001, 3:10–12.

C. Other Terms

As we do not perceive a need to construe other terms, we apply the plain and ordinary meaning to the remaining claim language.

IX. ANALYSIS

A. Legal Principles

“In an [inter partes review], the petitioner has the burden from the onset to show with particularity why the patent it challenges is unpatentable.” *Harmonic Inc. v. Avid Tech., Inc.*, 815 F.3d 1356, 1363 (Fed. Cir. 2016) (citing 35 U.S.C. § 312(a)(3) (requiring inter partes review petitions to identify “with particularity . . . the evidence that supports the grounds for the challenge to each claim”)). This burden of persuasion never shifts to patent owner. *See Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015) (discussing the burden of proof in inter partes review).

The question of obviousness is resolved on the basis of underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

Additionally, the obviousness inquiry typically requires an analysis of “whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007) (citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006) (requiring “articulated reasoning with some rational underpinning to support the legal conclusion of obviousness”)); see *In re Warsaw Orthopedic, Inc.*, 832 F.3d 1327, 1333 (Fed. Cir. 2016) (citing *DyStar Textilfarben GmbH & Co. Deutschland KG v. C. H. Patrick Co.*, 464 F.3d 1356, 1360 (Fed. Cir. 2006)).

An obviousness analysis “need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ.” *KSR*, 550 U.S. at 418; accord *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1259 (Fed. Cir. 2007). Petitioner cannot satisfy its burden of proving obviousness by employing “mere conclusory statements.” *In re Magnum Oil Tools Int’l, Ltd.*, 829 F.3d 1364, 1380 (Fed. Cir. 2016). Instead, Petitioner must articulate a reason why a person of ordinary skill in the art would have combined the prior art references. *In re NuVasive*, 842 F.3d 1376, 1382 (Fed. Cir. 2016).

A reason to combine or modify the prior art may be found explicitly or implicitly in market forces; design incentives; the “interrelated teachings of multiple patents”; “any need or problem known in the field of endeavor at the time of invention and addressed by the patent”; and the background knowledge, creativity, and common sense of the person of ordinary skill. *Perfect Web Techs., Inc. v. InfoUSA, Inc.*, 587 F.3d 1324, 1328–29 (Fed. Cir. 2009) (quoting *KSR*, 550 U.S. at 418–21).

In determining whether a claim is obvious in light of the prior art, when in evidence, we consider any relevant objective evidence of non-obviousness. *See Graham*, 383 U.S. at 17. Notwithstanding what the teachings of the prior art would have suggested to one of ordinary skill in the art at the time of the invention, the totality of the evidence submitted, including objective evidence of non-obviousness, may lead to a conclusion that the challenged claims would not have been obvious to one of ordinary skill. *In re Piasecki*, 745 F.2d 1468, 1471–72 (Fed. Cir. 1984). At this stage of the proceeding Patent Owner does not present evidence of such objective considerations.

We analyze the asserted grounds of unpatentability in accordance with these principles to determine whether Petitioner has met its burden to establish a reasonable likelihood of success at trial.

B. Petitioner’s Challenge to Claims 1–11 As Obvious Over Lindholm ’685 and Lindholm ’913 (collectively, “the Lindholm References”)

1. *Lindholm ’685 (Ex. 1005)*

Lindholm ’685 discloses a graphics processor that includes a multithreaded processing unit. Ex. 1005, 1:44–45.

The multithreaded processing unit includes a thread control unit configured to store pointers to program instructions associated with threads, each thread processing a sample type of vertex, pixel, or primitive. The multithreaded processing unit also includes at least one programmable computation unit configured to process data under control of the program instructions.

Id. at 1:45–51. Figure 2 of Lindholm ’685 is reproduced below.

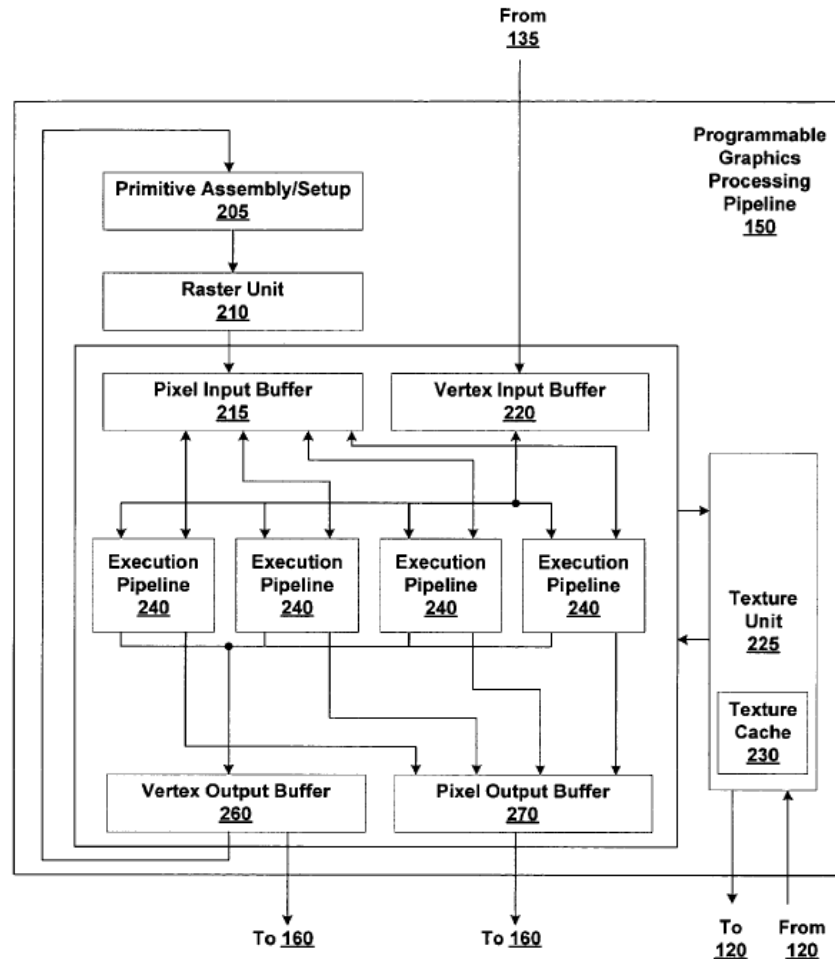


FIG. 2

Ex. 1005, Fig. 2. Figure 2 illustrates that Lindholm '685 includes execution pipelines 240, each containing a multithreaded processing unit that accepts and processes vertex and pixel samples from vertex and pixel input buffers 210, 220, respectively, when a thread is available. *Id.* at 4:31–36, 5:23–25. Figure 4 of Lindholm '685, showing details of a multithreaded processing unit, is reproduced below.

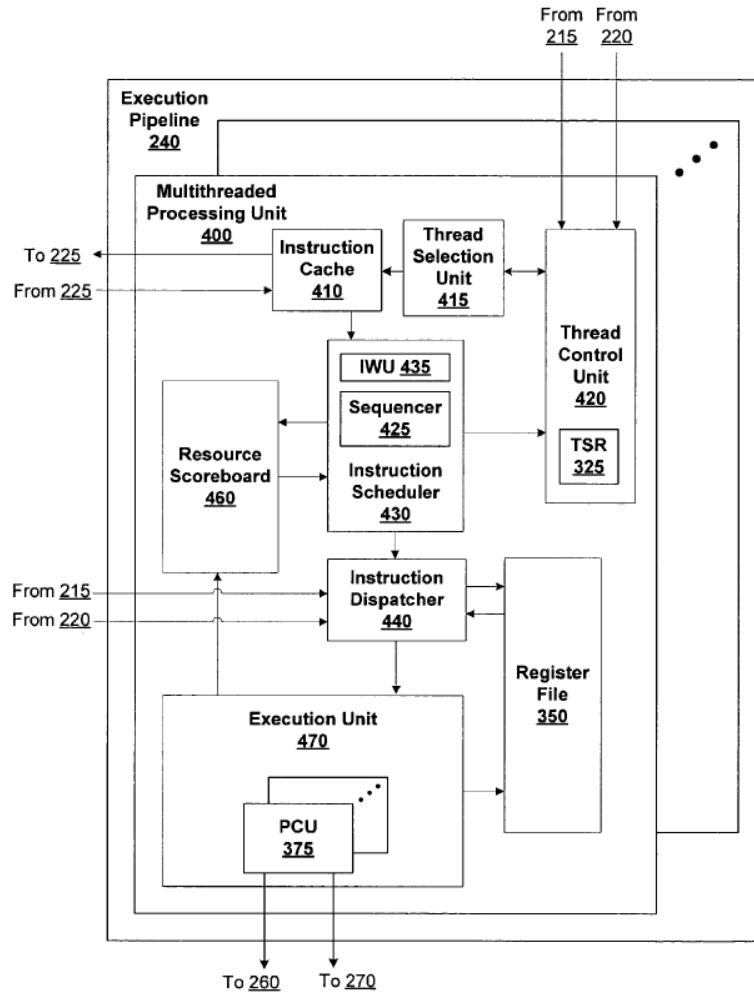


FIG. 4

Id., Fig. 4. Each thread processing unit can process vertex or pixel data—Instruction Dispatcher 440 gathers source data from pixel input buffer 215 and vertex input buffer 220 or Register file 350 specified in an instruction and outputs the instruction and source data to Execution unit 470. *Id.* at 9:34–38. When program instructions associated with a thread have completed execution, the storage resources allocated to retain intermediate data during thread execution become available to another thread. *Id.* at 9:57–63.

2. *Lindholm '913 (Ex. 1006)*

Lindholm '913 discloses a “programmable graphics processor that supports processing of graphics data elements in an order independent from the order in which the graphics data elements are received by the programmable graphics pipeline within the programmable graphics processor.” Ex. 1006, 1:39–44. The graphics processor includes at least one multithreaded processing unit that receives samples in a first order to be processed by program instructions associated with at least one thread. *Id.* at 1:48–52. Each multithreaded processing unit includes a scheduler. *Id.* at 1:53. The scheduler is configured to receive program instructions, determine the availability of source data, and schedule instructions for execution in a second order that is independent of the first order. *Id.* at 1:54–56. Each multithreaded processing unit includes a resource tracking unit that tracks the availability of source data, and a dispatcher configured to output program instructions in the second order to be executed by the multithreaded processing unit. *Id.* at 1:56–61. In one embodiment, when first source data required to process a program instruction associated with a first thread is not available, and second source data required to process a program instruction associated with a second thread are available, the program instructions associated with the second thread are dispatched prior to dispatching program instructions for the first thread. *Id.* at 2:7–22.

3. *Reasons to Combine the Teachings of Lindholm '685 and Lindholm '913*

Noting that Lindholm '685 and Lindholm '913 are both directed to a multithreaded shader and much of their Specifications and figures are the same, Petitioner contends that a person of ordinary skill would have recognized the Lindholm references disclose different details of the same

unified shader architecture. Pet. 21–22. Petitioner also states that an ordinarily skilled artisan would have looked to the additional disclosure in Lindholm ’913 for additional details about scheduling threads for execution independent of other threads. *Id.* at 22. We agree that a person of ordinary skill would have had reason to combine the teachings of the Lindholm references.

4. *Previous Litigation Background*

a) *Prosecution of the ’454 Patent*

The ’454 patent is a continuation of U.S. Patent No. 6,897,871. Ex. 1001, code (63). The parties do not dispute that the effective filing date of the challenged ’454 patent is November 20, 2003, and that the effective filing dates of Lindholm ’685 and Lindholm ’913 are earlier, i.e., June 30, 2003 and June 27, 2003, respectively. *See, e.g.*, Prelim. Resp. 24. Petitioner states that, during prosecution, the Examiner rejected the claims of the ’454 patent three times as anticipated by Lindholm ’685. Pet. 15. Petitioner emphasizes that Patent Owner never overcame the rejection on the merits, but instead filed a declaration under 37 C.F.R. § 1.131 antedating Lindholm ’685. Pet. 15.

Patent Owner contends that the Lindholm patents are not prior art because the inventors conceived of the inventions in the ’454 patent by August 24, 2001, while designing and developing the R400 graphics processor, and diligently reduced to practice until the ’454 patent’s effective filing date. Prelim. Resp. 24.

b) *Previous ITC Litigation*

Petitioner references an October 9, 2015 Final Initial Determination in ITC Inv. No 337-TA-932 (“the ’932 Investigation”). Pet. 15–16 (citing Ex. 1015, 140–145). Petitioner states that Administrative Law Judge

(“ALJ”) Pender found that the Respondents were unable to prove conception prior to the effective filing date of Lindholm ’685 for claims drawn to processing graphics and allocating threads by assigning priority to processing vertices or pixels, and claims drawn to determining whether to process pixel data or vertex data based on workload (load balancing). *Id.* (citing Ex. 1015, 140–145). Petitioner states that the ALJ also found Patent Owner’s documentary evidence of prior conception of its sequencer to be non-enabling and insufficient evidence of actual reduction to practice. *Id.* at 16 (citing Ex. 1015, 145–149).

Patent Owner notes that “the ’454 patent was not at issue in the ’932 Investigation, nor was conception and reduction to practice of its claims discussed in any way.” Prelim. Sur-reply 2. Patent Owner also states in the ITC Investigation

Judge Pender found requirements of Lindholm ’685—that are not present in the ’454 patent—were not shown in the ATI product prior to the relevant date, largely because the evidence was from 2002 onward and respondents sought a conception date in 2000. *Id.* at 141-145. The evidence here does not suffer from that flaw.

Id. at 8; *see also* Ex. 1015, 140–141 (for procedural reasons, limiting consideration of conception only to the R400 Architecture Proposal version 0.1, and finding “Respondents have waived any argument that asserts the ATI Unified Shader had a date of conception other than November 2000”).

c) Previous Federal Circuit Litigation Concerning Diligence To Reduce To Practice

As to diligence to reduction to practice, Patent Owner cites the court’s analysis in *ATI Techs. ULC v. Iancu*, 920 F.3d 1362, 1369 (Fed. Cir. 2019) concerning the ’871 patent, reversing a Board decision in IPR2015-00326

and finding Patent Owner's conception and diligence antedated the Lindholm references.

Petitioner responds that the Federal Circuit's decision concerns the claims of the parent '871 patent, where conception of the claimed subject matter was not contested, but that Patent Owner has not demonstrated conception of the subject matter recited in the '454 patent claims.⁸ Prelim. Reply 1–4. Petitioner argues that the '871 patent does not mention, and Patent Owner has not demonstrated conception of, the subject matter claimed in the '454 patent drawn to the allocation or use of register space as a predicate for determining whether to perform vertex or pixel operations. Prelim. Reply. 2–3 (referencing Ex. 1001, claim 1, limitations drawn to performing vertex operations unless the general purpose register block does not have enough available space to store incoming vertex data and continuing pixel operations until enough registers within the general purpose register block become available); *see also* Pet. 16, n. 1. According to Petitioner,

[t]his is a critical distinction because [Patent Owner] did not conceive of the use of register allocation in the context of the R400 or any other project, as recited in the Challenged Claims, prior to their effective filing date. Patent Owner failed to address this distinguishing factor in its Preliminary Reply. At most, [Patent Owner] identified a problem, as confirmed by its witness testimony, but it did not identify a solution, which is required to establish prior inventorship.

Id. at 3.

⁸ The Federal Circuit case also concerned diligence and reduction to practice of the claims of U.S. Patent No. 7,329,369 (also a parent of the '454 patent) and U.S. Patent No. 7,742,053, addressed in IPR2015-00330 and IPR2015-00325, respectively

5. *Analysis of Whether Patent Owner Antedated the Lindholm References As Applied To The '454 Patent Claims*

As the claims of the '454 patent were not the subject of either the decision by the Federal Circuit or the ITC Investigation, we address the issue of whether Patent Owner has antedated the Lindholm references based of the evidence of record before us.

“The test for conception is whether the inventor had an idea that was definite and permanent enough that one skilled in the art could understand the invention.” *Tyco Healthcare Grp. LP v. Ethicon Endo-Surgery, Inc.*, 774 F.3d 968, 974–975 (Fed. Cir. 2014). As to conception of the subject matter claimed in the '454 patent, Patent Owner cites (i) the R400 Top Level Specification dated March 14, 2001 authored by inventor Steve Morein (Ex. 2103), (ii) the R400 Shader Processor Specification dated January 23, 2001 authored by inventors Andrew Gruber and Andi Skende (Ex. 2012), and (iii) the R400 Sequencer Specification dated August 24, 2001 authored by inventor Laurent Lefebvre (Ex. 2010). Prelim. Resp. 25–30. According to Patent Owner the R400 Top Level Specification describes the R400 as including a shader processor and a sequencer, and explains that the R400 implements a unified processing pipe that includes a processor unit to perform 3D vertex and pixel operations. *Id.* at 28–29 (citing Ex. 2013, 9–10; Ex. 2001, Declaration of Dr. William Mangione-Smith (“Mangione-Smith Decl.”) ¶¶ 79, 98, 121, 133, 145, 182). Patent Owner cites the R400 Shader Processor Specification as describing general purpose registers configured to store both vertex and pixel data. *Id.* at 29 (citing Ex. 2001, Mangione-Smith Decl. ¶¶ 103, 125, 137).

According to Patent Owner, the R400 Top Level Specification and the R400 Sequencer Specification describe a sequencer having the same

characteristics as the sequencer claimed in the '454 patent. *Id.* at 30. Patent Owner contends these documents describe that, as part of the rasterizer, the R400 sequencer performs arbitration, is coupled to the general purpose register block and processor unit, and maintains instructions operative to cause the processor to execute vertex and pixel calculation operations based on selected data maintained in the general purpose register block. *Id.* (citing Ex. 2010, 12; Ex. 2013, 11; Ex. 2001, Mangione-Smith Decl. ¶¶ 76, 82, 85, 92, 105–111, 152).

Patent Owner also emphasizes that a feature of the '454 patent claims is the determination of what data to process by evaluating dynamic storage capacity. Prelim. Resp. 13; Ex. 1001, 6:52–64 (claim 1), 7:12–8:21 (claims 3–10). Patent Owner states,

[i]n contrast to prior approaches, including that of the prior art asserted here, which may use pre-determined priorities or balance workloads based on how many of a particular task is in the queue to execute, the system of the '454 patent selects the specific data based on the availability of space in the data storage for the particular types of data.

Prelim. Resp. 13 (citing Ex. 2001, Mangione-Smith Decl. ¶ 38); *see* Ex. 1001, 2:58–3:17, 5:32–44 (“[I]f the general purpose register block does not have enough available space therein to store the incoming vertex data, such information will not be transmitted as the arbitration scheme of arbiter 64 is not satisfied.”); *see also* claims 1, 3–10 (reciting continuing pixel or vertex operations until sufficient shared resources, e.g., space available in a data store or registers in general purpose register, become available).

In support of its conception of this subject matter, Patent Owner quotes the R400 technical specifications beginning in August 2001 as stating (i) the sequencer of the R400 “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” (Ex. 2010⁹, 18), (ii) “the sequencer will not start the next vector until the needed space is available” (Ex. 2010, 4 (referring to the number of general purpose registers (GPRs) needed to execute)), (iii) “the arbiter is not going to select a vector to be transformed if the parameter cache is full” (Ex. 2010, 17), and (iv) “there are actually two sets of arbitration - - one for pixels and one for vertices ALU arbitrations is a little more complicated these threads are further filtered based on whether space is available” (Ex. 2011¹⁰, 26). Prelim. Sur-reply 3 (emphasis by Patent Owner); *see also* Ex. 2010, 3 (Laurent Lefebvre entry stating “[r]eviewed the Sequencer Spec after meeting on August 3, 2001. Added dynamic allocation for register file and example (written in part in Vic) of the flow of pixels/vertices in the sequencer.”)

Patent Owner also cites the Sequencer Specification dated August 29, 2003 (Ex. 2119, 1281–1282 (repeating the same text as that cited from the 2001 Top Level Specification for the R400)), the Mangione-Smith Declaration paragraphs 8, 127, 139 (all citing Ex. 2013¹¹, 9–10 as disclosing

⁹ Calvin Watson identifies Ex. 2010 as the R 400 Sequencer Specification ver. 0.4 and states that it was revised and checked in by Laurent Lefebvre on August 24, 2001. Ex. 2003, Watson Decl. ¶¶ 33–34.

¹⁰ Calvin Watson authenticates Ex. 2011 as the R 400 Sequencer Specification ver. 0.5 and states that it was revised and checked in by Laurent Lefebvre on September 7, 2001. *See* Ex. 2003, Watson Decl. ¶¶ 33–34.

¹¹ Dr. Mangione-Smith identifies Ex. 2013 as “R400 Top Level Specification (Version 0 2). March 11, 2001. *See* Ex. 2001, Mangione-Smith Decl. at iii. Calvin Watson authenticates Ex. 2013 as the R400 Sequencer Specification, ver. 0.7 and states that Laurent Lefebvre revised and submitted this document, on Oct. 5, 2001. *See* Ex. 2003, Watson Decl. ¶¶ 42–43. In the record, Exhibit 2013 appears to be an R400 Top Level

“the number of registers required for an operation are specified when the task is submitted to the unified shader. The unified shader will not start the task until there is enough free room for the task’s registers”), and the Declaration of Calvin Watson (Ex. 2003, Watson Decl. ¶¶ 33–35 (authenticating Ex. 2010), 85–87 (authenticating Ex. 2028 as R400 Sequencer Specification (version 2.1), revised and checked in by Laurent Lefebvre on April 19, 2002), 118–120 (authenticating Ex. 2039 as the R400 Sequencer Specification (Version 2.11, revised and checked in several times by Laurent Lefebvre between May 1, 2003 and August 29, 2003))). Prelim. Sur-reply 4.

The cited material from the August 2001 Top Level Specification for the R400 at least suggests conception of an idea that was definite and permanent enough that one skilled in the art could understand the invention, i.e., using available register space as a predicate for performing vertex or pixel processing.

Petitioner attacks Patent Owner’s conception contentions by arguing that Dr. Mangione-Smith relies only on draft specifications for the R400 and ignores other evidence. Pet. Reply 4. Petitioner points out that Dr. Mangione-Smith testified that the ATI inventors created an implementation of the R400 in source code, but that Dr. Mangione Smith did not analyze that source code. *Id.* Petitioner argues that Dr. Mangione-Smith’s failure to

Specification ver. 0.2 with an Originate Date of 11 March 2001, an Edit Date of 4 Sept. 2015, and states “This replaces the R400 architecture specification.” *See* Ex. 2013, 1. The “revision changes” in Ex. 2013 refer to Rev. 0.0 (Steve Morein) dated March 11, 2001 Initial Revision and include the notations “Document recreated from earlier documents; Finally got back to editing it.” In the absence of further information concerning its content and origins, Exhibit 2013 has not been authenticated.

address the RTL (register-transfer level) code is a significant omission, given the testimony of Patent Owner's witness Andrew Gruber that "RTL is probably the best guide" and "anything not in the RTL code is not in the GPU (graphics processing unit)." *Id.* (citing Ex. 1017, Transcript of Deposition of Andrew Gruber ("Gruber Tr.") 52:11–53:3). Petitioner also cites the testimony of Dr. Andrew Wolfe, who reviewed that RTL code, and stated "The RTL code is the evidence of actual reduction to practice. The C++ code is – basically corroborates it." (Ex. 2007, Transcript of Deposition of Dr. Andrew Wolfe ("Wolfe Depo. Tr.") 304: 11–13, 305:2–5). Although Petitioner does not cite record evidence, according to Petitioner, Dr. Wolfe concluded that the R400 had a fixed vertex priority scheme that did not depend on any amount of data in storage, a system Petitioner characterizes as exactly opposite to that claimed in the '454 patent. Pet. Sur-reply 4: *see e.g.*, Ex. 2002, Declaration of Dr. Andrew Wolfe ("Wolfe Decl.") ¶¶ 174–198 (discussing alternatives as to how the R400 Emulator code describes an arbiter operable to select a command thread based on relative priorities of pixel and vertex command threads).

According to Petitioner, Mr. Andrew Gruber testified that in the static mode the R400 did not utilize any priority and could not utilize any allocation of shared resources or give priority to vertex or pixel threads, as required by the challenged claims. Pet. Reply 8 (citing Ex. 1007, Gruber Tr. 167:13–21 (answering "no, it does not," when asked if the GPR allocation module uses priority in the static mode)). Arguing that Mr. Gruber's testimony contradicts that of Dr. Mangione-Smith, Petitioner contends that Patent Owner cannot show conception or diligence to reduce to practice of the subject matter claimed in the '454 patent. *Id.* However, although Mr. Gruber did not identify specific code, when asked how it is determined

whether to allocate more space for pixels or vertices when arbitrating between pixel and vertex threads, Mr. Gruber testified that he knows from the high level documentation “it uses the amount of space in the output buffers to decide whether we’re more pixel limited or more vertex limited.” Ex. 1017, Gruber Tr. 167:3–12.

Patent Owner argues there is substantial evidence of conception of the claimed subject matter and due diligence regarding the R400 sequencer. Prelim. Sur-reply 4. According to Patent Owner, Mr. Gruber’s testimony demonstrates that the basic design was proven to work, even if it had “bugs” and was not completely functional. Ex. 1017, Gruber Tr. 103:5–21 (stating the dynamic mode of the register file allocation method of the R400 “was basically functional” but may have “had some bugs associated with it” and “[t]he basic design was proven to work”). According to Mr. Gruber, “the nature of these bugs were subtle enough that they would not have come up until you had silicon. So it wouldn’t have been until the R600 . . . there may have been an issue with the Xenos silicon with this [dynamic register file allocation].” Ex. 1017, Gruber Tr. 103:24–104:9. Mr. Gruber also acknowledged that when the first silicon was made for the Xenos chip, the dynamic register file allocation method did not function properly. Ex. 1017, Gruber Tr. 104:20–25. Mr. Gruber further testified that, when he signed the Rule 1.131 Declaration to antedate the references asserted against the patents in the ’932 Investigation, “[m]y personal understanding was the model code and the chip design code demonstrated a functional embodiment of the patent and that the model code and the RTL codes were codes from the R400 design effort prior to June, 2003 - - and I knew we had such codes running functionally at that time.” Ex. 1017, Gruber Tr. 124:22–125:9.

Petitioner emphasizes Mr. Gruber’s testimony stating the dynamic register allocation included flaws that prevented it from being functional and may not have come close to working in any form until the R600 device was developed years after the patent filings. Prelim. Reply 7 (citing Ex. 1017, Gruber Tr. 104:20–25, 104–107, 107:10–16 (stating that R400 and the Xenos design (launched in the Microsoft Xbox in November 2005¹²) used a fixed register space widely available at the time)). Petitioner further points out that the R400 was never taped out and the dynamic register allocation never worked. *Id.* at 5, 7 (citing Ex. 1017, Gruber Tr. 104:20–25 (discussing the post R400 Xenos design as first taped out in late 2004–2005)). Patent Owner argues that proving conception and diligence to reduce to practice does not require a taping out of an actual graphics processing unit (GPU). Prelim. Sur-reply 5.

The subject matter of the ’454 patent claims was not the subject matter of the claims addressed in previous Federal Circuit litigation, where conception was not at issue. *ATI Techs. ULC v. Iancu*, 920 F.3d 1362, 1369 (Fed. Cir. 2019). As discussed above, the August 2001 Top Level Specification for the R400 indicates that the inventors conceived of using available register space as a predicate for performing vertex or pixel processing in a manner that was sufficiently definite that one skilled in the art could understand the invention. *See* Ex. 2010, 4, 17, 18. As this subject matter of the ’454 patent claims was conceived of and developed as part of the R400 project, we consider relevant aspects of the Federal Circuit’s conclusion that Patent Owner demonstrated sufficient diligence to reduce the invention at issue in that case to practice. The Court noted that “[d]iligence

¹² *See* Ex. 1015, 139.

is not negated if the inventor works on improvements and evaluates alternatives while developing an invention.” *ATI Techs. ULC*, 920 F.3d at 1372 (citing *In re Jolley*, 308 F.3d 1317, 1328 (Fed. Cir. 2002) (declining to adopt a rule that evidence of diligence must be excluded if there is any possibility that it could be construed in support of an invention beyond the reach of the count)). The court found “no basis for finding that this technology was not being diligently pursued.” *Id.* at 1373. The court also rejected arguments that “ATI could have filed its patent applications at an earlier stage of development and that continuing activity to develop the products does not count as diligence,” stating “[p]recedent is not in accord; the purpose of the diligence requirement is to show the invention was not abandoned or set aside.” *Id.*

Petitioner appears to be arguing that Patent Owner had not conceived of the claimed subject matter because it was unable to obtain a working version at least until introduction of the R600 shader. That is not the test for conception. *See Tyco Healthcare*, 774 F.3d at 974–975 (“The test for conception is whether the inventor had an idea that was definite and permanent enough that one skilled in the art could understand the invention.”). Even if Petitioner is correct that the dynamic memory aspects of the R400 design did not work properly or were not embodied into a fully operable product until the R600 shader, the documentary evidence indicates that Patent Owner conceived of the claimed subject matter in the early stages of R400 development and did not abandon or set aside that aspect of the technology. Thus, for purposes of this Decision and on the preliminary record, we are persuaded that Patent Owner has antedated the Lindholm references.

6. *Petitioner's Contentions Concerning the Lindholm Technology*

Recognizing that our analysis concluding that Patent Owner has antedated the Lindholm references is subject to further development of the evidence at trial, we provide additional analysis of the merits of Petitioner's challenge.

a) *Petitioner's Contentions - Claim 2*

Petitioner contends that the Lindholm references disclose a multithreaded unified shader (claim 2 preamble) that includes a plurality of pipelines each capable of receiving vertex data from vertex input buffer 220 and pixel data from pixel buffer 215. Pet. 25–26 (citing Ex. 1005, Fig. 2, 4:22–5:35; Ex. 1006, Fig. 2, 4:35–5:45; Ex. 1003, Pfister Decl. ¶ 89). Petitioner notes that each execution pipeline can perform vertex operations, e.g., transforming objects from object space to another coordinate system, and pixel operations, e.g., texture mapping, shading, and blending. *Id.* at 24–25 (citing Ex. 1005, 4:42–5:10).

Petitioner further cites Lindholm '685 as disclosing a general purpose register block (limitation 2[a]), in the form of Register File 350 that maintains source data, including vertex and pixel data, as well as pixel input buffer 215 and vertex input buffer 220 that provide vertex and pixel data to thread control unit 420. Pet. 25–26 (citing Ex. 1005, 5:50–63). Petitioner notes that an ordinarily skilled artisan would have understood the pixel and vertex input buffer, to include a register file, FIFO, cache and the like, to be portions of a general purpose register. *Id.*

Petitioner designates as claim limitations 2[b] and 2[d] the processor unit that executes instructions that generate a pixel color in response to selected data from the general purpose register and vertex and position and

appearance data in response to selected data for the general purpose register.

Pet. 28. Petitioner cites Execution Units 470 and programmable computation units (“PCUs”) as operatively coupled to Register Files 350 and executing instructions that perform pixel operations, e.g., instructions that generate pixel color, and perform vertex operations, e.g., instructions that generate vertex position and appearance data. *Id.* at 28–29. Petitioner states that the data received by the Execution Units and PCUs is “selected” for processing by the Thread Selection Unit, according to thread allocation priorities and execution priorities. *Id.* at 29 (citing Ex. 1005, 6:61–7:36, 7:56–8:32; Ex. 1003, Pfister Decl. ¶ 100).

As to the claimed sequencer (limitation 2[c]), Petitioner cites the combination of Instruction Cache 410, Instruction Scheduler 430 (which includes Sequencer 425 and IWU 435), and Instruction Dispatcher 440. Pet. 29 (citing Ex. 1003, Pfister Decl. ¶ 102). Petitioner states these components function together and maintain instructions that cause the Execution and PCU units to execute vertex and pixel operations on selected data maintained in the general purpose register block. *Id.* Noting that the Instruction Cache outputs program instructions to the Instruction Scheduler 430, Petitioner states that Instruction Scheduler 430 stores a predetermined number of instructions and evaluates whether the Execution Units/PCUs can execute an instruction based on the availability of computation resources and source data in Register File 350. *Id.* (citing Ex. 1005, 8:47–9:4). Upon the Instruction Scheduler determining an instruction and associated thread will be executed, Instruction Dispatcher 440 gathers the source data from the pixel input buffer, vertex input buffer or Register File, and outputs the instruction and source data to the Execution Unit. *Id.* at 29–30 (citing Ex. 1005, 9:33–49, Fig. 4).

b) Petitioner's Contentions - Claims 1, 3, 4, 5, and 11

As to all claims, Petitioner contends that the Lindholm references disclose a unified shader for the same reasons as those discussed with respect to claim 2. Pet. 30¹³.

As to limitation 3[b], Petitioner references its discussion of limitation 2[a] and identifies the claimed shared resources in the '454 patent as components that store both vertex and pixel data. *Id.* at 35. Petitioner cites Register File 350, as well as Pixel Input Buffer 215 and Pixel Input Buffer 220 in the Lindholm references as disclosing this limitation. *Id.* Portioner notes that each of these buffers is coupled to the Execution Units/PCUs through Instruction Dispatcher 440, which gathers vertex and pixel data from those memory locations and outputs it to the Execution Units/PCUs for their use. *Id.*

Petitioner designates claim limitation 3[a] as reciting a processor unit operative to perform vertex and pixel calculation operations. Pet. 30. Petitioner designates claim limitation 3[c] as reciting that the processor unit uses shared resources for either vertex data or pixel information and “is operative to perform pixel calculation operations until enough shared resources become available and then use the shared resource to perform vertex calculation operations.” *Id.* Petitioner address these limitations together. Petitioner cites the Lindholm references as disclosing that both vertex and pixel operations are performed simultaneously within each multithreaded processing unit and data is assigned based on a thread

¹³ Petitioner discusses the preamble of claim 2 in the Petition at section VII.D.4.a. Pet. 23. We understand Petitioner's references to §2[preamble], and subsequent references using the § sign to reference corresponding claim limitations, not sections of the Petition.

allocation priority for each sample type. *Id.* at 31 (citing Ex. 1005, 6:50–7:36; Ex. 1006, 7:43–58). According to Petitioner, in a fixed vertex thread allocation, vertex operations are performed until sufficient shared resources become available to begin pixel calculation. *Id.* (citing Ex. 1003, Pfister Decl. ¶¶ 107–108; Ex. 1005, 6:60–7:5). Similarly, a fixed pixel thread priority would give priority to pixel threads, and vertex operations would be performed only if no pixels samples were available for thread assignment. Ex. 1005, 10:17–54, 16:10–55. Petitioner also points out that Lindholm can apply priority based on vertex or pixel workload, disclosing, e.g., that “[d]ynamically modifying a thread allocation priority for pixel samples based on the amount of sample data in Pixel Input Buffer 215 permits Pixel Input Buffer 215 to drain faster.” Pet. 32 (citing Ex. 1005, 7:6–36 (providing a similar disclosure for prioritizing vertex samples)). Petitioner further points out that the processing priority for vertices or pixels can be set or changed based on the availability of resources. *Id.* at 33 (citing Ex. 1005, 7:36–40 (“[o]nce a thread is assigned to a source sample, the thread is allocated storage resources, such as locations in Register File 350”), 12:9–12 (the maximum number of threads that can be executed simultaneously is related to . . . the size of storage for thread state data”), 16:19–24 (“thread execution priority is determined based on an amount of sample data in Pixel Input Buffer 215 and another amount of sample data in Vertex Input Buffer 220 and optionally stored in the thread execution priority register”).

Petitioner contends that the processing units in claims 1, 3, 4, 5, and 11 are disclosed or rendered obvious by the Lindholm references because they disclose simultaneously processing vertices and pixels based on the workload and availability of resources at various degrees of completion (explicitly recited in claim 11). Pet. 34–35 (citing Ex. 1006, 7:19–10:3,

13:17–14:62). Referring to the disclosure in Lindholm ’913 that the Instruction Scheduler can output multiple instructions per clock cycle corresponding to multiple threads, and the execution units are able to simultaneously process multiple samples, Petitioner contends that the execution units can freely switch back and forth between vertex and pixel operations to ensure the graphics processor is dynamically balanced. *Id.* at 34–35 (citing Ex. 1006, 13:24–39; Ex. 1003, Pfister Decl. ¶¶ 115–118).

c) Petitioner’s Contentions - Claim 6

Claim 6 depends from independent claim 5 and recites that the sequencer includes circuitry operative to fetch data from a memory. Ex. 1001, 8:9–10. Petitioner cites Instruction Dispatcher 440 in Lindholm ’685 as disclosing such circuitry, noting that the Instruction Dispatcher gathers source data from Pixel Input Buffer 215, Vertex Input Buffer 220 or Register File 350. Pet. 37 (citing Ex. 1005, 9:33–49, 11:9–14, 15:24–30).

d) Petitioner’s Contentions - Claims 7 and 10

Claim 7 depends from claim 5 and recites that the shader further includes a selection circuit operative to provide information to the store in response to a control signal. Ex. 1001, 8:11–13. Claim 10 depends from claim 7 and recites that the control signal is provided by an arbiter. *Id.* at 8:20–21. Referencing Lindholm ’685, Petitioner cites Thread Selection Unit 415 as disclosing the claimed selection circuit and Thread Control Unit 420 as disclosing the claimed arbiter. Pet. 37. Petitioner explains that the Thread Selection Unit reads thread entries including thread state data from the Thread Control Units which assigns threads or samples to threads based on thread allocation or priority. *Id.* at 37–38 (citing Ex. 1005, 6:60–7:5, 7:6–8:17, 15:14–17:7, 16:10–55, Fig. 3–4). According to Petitioner, to the extent it is not explicit, it would have been obvious to provide the priority

control signal from the Thread Control Unit to the Thread Selection Unit. *Id.* at 38 (citing Ex. 1003, Pfister Decl. ¶ 131). Petitioner notes that Lindholm's Thread Selection Unit provides information to the Register File 350, i.e., the claimed store, at least through the threads selected for execution, using the Instruction Cache, Instruction Scheduler, and Instruction Dispatcher, as well as the Execution Unit and as a result of the threads selected for execution. *Id.* (citing Ex. 1005, 8:15–9:67, 5:50–6:3, 7:37–40, 8:47–9:4, 9:33–66).

e) Petitioner's Contentions – Claim 8 and 9

Claim 8 depends from claim 5 and recites that the processor unit executes instructions that generate a pixel color in response to the selected data. Ex. 1001, 8:14–16. Claim 9 depends from claim 5 and recites that the processor unit generates vertex position and appearance data in response to selected data. *Id.* at 8:17–19.

As to the limitations in claims 8 and 9, Petitioner references its discussion of limitation 2[d]. Pet. 39.

7. Patent Owner's Contentions and Analysis

Patent Owner responds to Petitioner's contentions by addressing specific groups of claims and certain claim limitations. Recognizing that Petitioner bears the burden of persuasion as to all challenges, we address Patent Owner's contentions and provide our analysis below.

a) All claims

Patent Owner contends that the Lindholm references do not teach a unified shader because they do not teach switching between unfinished vertex and pixel threads. Prelim. Resp. 39–40.

Patent Owner's arguments, which are based on its definition of a unified shader, are unavailing, as we do not construe unified shader to

include the features discussed by Patent Owner. *See* Section VIII. B herein. Consistent with the '454 patent specification, we construe a unified shader to mean a shader that is configured to perform both vertex and pixel operations. *See id.*

b) Claim 2

As to claim 2, Patent Owner contends Lindholm '685 does not disclose that the receipt of data causes execution of instructions. Prelim. Resp. 43. Noting that the Lindholm subject matter cited by Petitioner states that “[p]rogram instructions configure programmable computation units (PDU)s within Execution Pipeline 240 to perform operations,” Patent Owner argues that Lindholm teaches the instruction scheduler determines which instructions and threads will be executed. Prelim. Resp. 43 (citing Ex. 1005, 5:23–25, 9:5–7). Patent Owner’s argument is unavailing. As Petitioner points out, Lindholm discloses embodiments in which locations in Register File 350 are assigned to threads based on priority assigned to each sample type. *See id.* at 29; *see also* Ex. 1005, 6:60–7:36 (describing alternative priority arrangements for thread execution depending on sample types).

c) Claims 1, 3–11

Patent Owner contends that Petitioner has not demonstrated that either of the Lindholm references discloses considering the amount of space available in the store or how storage capacity impacts the processing of particular data. Prelim. Resp. 40–41 (citing Ex. 2001, Mangione-Smith Decl. ¶¶ 312–313). According to Patent Owner the subject matter cited in the Petition relates to a priority scheme based on data type and tracking which processors are free to process tasks, i.e., whether processors are busy, but does not consider data storage capacity. *Id.* at 41. This argument is unavailing. As Petitioner points out, 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. Pet. 33; Ex. 1005, 12:9–12, 16:19–24.

d) Conclusion

As discussed above, based on the current record, we are persuaded that Patent Owner has antedated the Lindholm references and therefore they are not prior art. Therefore, we are not persuaded that Petitioner will succeed on its challenge based on the Lindholm references.

If the Lindholm references were considered applicable prior art, we are persuaded that Petitioner has cited sufficient evidence to demonstrate that a person of ordinary skill would have been motivated to combine the teachings of Lindholm '685 and Lindholm '913, and that in combination, the references would have taught, or at least suggested, all the limitations of claims 1–11 to a person of ordinary skill in the art.

C. Petitioner's Challenge to Claims 1–11 As Obvious Over Amanatides and Kohn

1. Amanatides (Ex. 1007)

Amanatides discloses a graphics architecture in which a host sends graphics primitives to a graphics pipeline front end. Ex. 1007, 156, Fig. 1. The front end performs geometric (vertex) operations (G) using floating-point ALUs. *Id.* Transformed/shaded primitives are sent to a back end that performs rasterization and visibility operations (R). *Id.* Amanatides proposes an architecture, as shown in Figure 3 reproduced below.

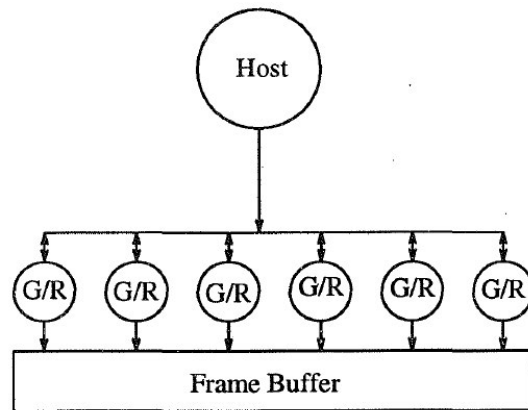


Figure 3

Ex. 1007, Fig. 3. In Amanatides' architecture, the functions of front ends (G) and back ends (R) are collapsed onto the same processors (G/R). *Id.* at 157. Each processor performs front end tasks, e.g., transforming/clipping/lighting/perspective operations, for the primitives and broadcasts the results to the other processors to perform packet and rasterization operations. *Id.* According to Amanatides, the design is "auto load balancing," such that, if there is a lot of texture mapping, the processors spend most of their time on that, but if most of the primitives are simple, the processors spend the majority of their time on the front end computation. *Id.* Amanatides notes that the approach employs powerful general purpose processors, instead of special purpose graphics hardware and, because the hardware pipeline is not optimized for a particular algorithm, capabilities can be changed and enhancements can be made freely. *Id.*

Figure 4, reproduced below, is a general outline of the design proposed by Amanatides, as implemented with Intel i680 general purpose microprocessors optimized for graphics. *Id.* at 158.

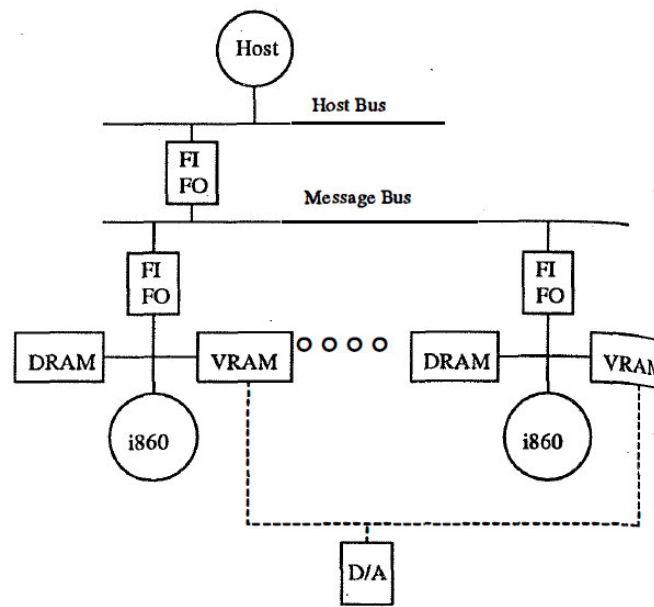


Figure 4

Id. Fig. 4. In Figure 4, processors are connected to each other and the host via a bidirectional FIFO connected to a message bus that handles both high and low priority packets. *Id.* at 159. The FIFOs help keep each processor busy and help perform the necessary multicasting without processor intervention. *Id.* Each processor waits for packets in its input FIFO. *Id.* Packets containing polygons can be geometry messages (GM) and rendering messages (RM). *Id.* GMs come from the host—the polygons are transformed, clipped, shaded, and then sent out as RMs via an output FIFO to the other processors. *Id.* When a processor receives an RM, the processor scan converts the polygon and draws it in its frame buffer. *Id.*

GMs are treated as low priority packets that cannot access the bus if high priority packets, i.e., RMs, are waiting. *Id.* A worst case scenario occurs when every input queue is full of GMs. *Id.* As a processor turns one of these GMs into an RM, it shuts down further introduction of GMs from the host. *Id.* at 160. As GMs turn into RMs, they wait in output FIFOs. *Id.* Whenever space appears at any input FIFO, a high priority RM is delivered

to it. Eventually all the GMs are turned into RMs and delivered to FIFOs. *Id.* When all the output FIFOs are empty and the processors are working on the remaining RMs in their input FIFOs, the host can begin to deliver GMs again. *Id.* Eventually, oscillating between working on GMs and RMs, each processor is doing useful work. *Id.*

2. *Kohn (Ex. 1008)*

Kohn discloses that the i860 CPU microprocessor, a 64-bit Reduced Instruction Set Computer (RISC) has chip area divided evenly between blocks for integer operations, floating point operations, and instruction and data cache memories. Ex. 1008, 15. The i860 contains separate register files for integer and floating point units to support parallel execution. *Id.* at 16. In addition, there are six control registers and four special purpose registers. *Id.* The i860 operates on standard integer and floating point data, as well as pixel data formats for graphics operations. *Id.* The floating point hardware performs transformation calculations and advanced lighting calculations required for 3D graphics. *Id.* at 24. Kohn also describes the basis of graphics processing, noting that in 3D graphics, polygons represent the points on the surface of a solid object, that the graphics unit calculates only the vertices of the polygons, that the unit knows the colors of the polygons and must calculate the points, i.e., the pixels, between the vertices and that intensity interpolation provided smoother linear changes in pixel intensity and color between vertices than flat shading of the polygons. *Id.*

3. *Reasons To Combine The Teachings of Amanatides and Kohn*

Petitioner states that a person of ordinary skill in the art would have had reason to combine Amanatides with Kohn because both references use the Intel i860 processor and both references describe processing graphics

with that same processor. Pet. 44 (citing Ex. 1003, Pfister Decl. ¶ 165). Petitioner notes that Amanatides states “[a]fter exploring several DSP microprocessors, we decided on the Intel i860. It is a powerful general-purpose microprocessor optimized for graphics.” *Id.* at 45 (citing Ex. 1007, 4). Petitioner also notes that Kohn states that the designers of the i860 processor “recognized a synergistic opportunity to incorporate a 3D graphics unit that supports interactive visualization of results.” *Id.* (citing Ex. 1008, 15).

In view of these disclosures, we agree with Petitioner that an ordinarily skilled artisan would have recognized that the i860 processor would be useful to implement the graphics architecture described in Amanatides and would have had reason to combine the teachings of the references.

4. *Petitioner’s Contentions – Claim 2*

Petitioner first addresses the limitations of illustrative claim 2. Pet. 45–52.

a) Claim 2 Preamble

Petitioner cites Amanatides alone or in combination with Kohn as disclosing a unified shader, as recited in the preamble of claim 2. Pet. 45. Petitioner cites Amanatides as disclosing a unified shader because vertex processing (G) and pixel processing (R) may occur in the same processor. Pet. 45–46 (citing Ex. 1007, 157; Ex. 1003, Pfister Decl. ¶ 169). Although we do not construe the term “unified shader” to require a shader that can switch back and forth between executing *unfinished* vertex and pixel threads, as urged by Patent Owner, Petitioner’s expert, Dr. Pfister, notes that Kohn observes that floating point and intergroup processing units in the i860 processor consist of multiple stages, implying that different operations

selected for a pipeline may be at different stages of execution. Ex. 1003, ¶ 183 (citing Ex. 1008, 22–23). Noting that Kohn describes parallel execution of operations in a “dual-operation mode” within the “pipelined mode,” where both floating point operations and scalar operations could take advantage of the pipelining capability, Dr. Pfister testifies that an ordinarily skilled artisan would comprehend, based on a combination of Amanatides and Kohn, that the functional units within the i860 processor can concurrently perform vertex operations and pixel operations, potentially at different stages of completion, by switching between instructions stored in the instruction cache. *Id.*

b) Limitation 2[a]

Petitioner cites Amanatides as disclosing the general purpose register block recited in claim limitation 2[a]. Pet. 46–48. Petitioner cites Amanatides as having an input FIFO for each processor that stores both vertex and pixel data, a DRAM coupled to the processor storing vertex and pixel data, a VRAM associated with the i860 processor, and an on-chip cache within each i860 processor that stores vertex and pixel data. *Id.* at 46–47 (citing Ex. 1007, 158–159; Ex. 1008, 15; Ex. 1003, Pfister Decl. ¶ 170). Petitioner states that these components individually and collectively constitute the claimed general purpose register block recited in claims 1 and 2, as well as the data store of claim 5. *Id.* at 48.

c) Limitations 2[b] and 2[d]

Petitioner contends that the claimed processor unit can be read on each G/R shown in Amanatides and on the functional units within the i860 processor, i.e., the floating point control unit, adder unit, and multiplier unit together with the 3D graphics integer unit, collectively or individually, each

of which performs both vertex and pixel operations. Pet. 48 (citing Ex. 1008, 15, 20, 24).

Petitioner argues that in each of Amanatides' G/R processors, the front end geometry (vertex processing) occurs simultaneously with back end pixel or rendering operations. *Id.* at 49 (citing Ex. 1007, 156–157, Fig. 3; Ex. 1003, Pfister Decl. ¶173). Petitioner notes that the '454 patent relies on partitioned sections or stages of a pipelined processor to perform simultaneous processing of vertex and pixel data. *Id.* at 50 (citing Ex. 1003, Pfister Decl. ¶174). Petitioner cites Kohn as disclosing that the i860 processor used by Amanatides includes floating point units that perform vertex operations and support scalar operations typical of pixel operations. *Id.* Petitioner further argues that Amanatides' load balancing architecture results in the i860 operating as a unified shader that switches between vertex and pixel processing depending on the type of data present in the registers. *Id.* at 50–51 (citing Ex. 1007, 157–158; Ex. 1003, Pfister Decl. ¶175). According to Petitioner, Amanatides also discloses that geometry/vertex operations are performed by its processors in response to receiving geometry messages with vertex data (GMs) and pixel operations are performed in response to receiving rendering messages with pixel data. *Id.* at 51 (citing Ex. 1007, 159–160; Ex 1003, Pfister Decl. ¶177).

d) Limitation 2[c]

Claim limitation 2[c] recites “a sequencer, coupled to the general purpose register block and the processor unit, the sequencer maintaining instructions operative to cause the processor unit to execute vertex calculation and pixel calculation operations on selected data maintained in the general purpose register block.” Petitioner cites the on-chip cache, together with Memory Management Unit in the i860 discussed by Kohn as

describing the claimed sequencer and sequencer circuitry. Pet. 51–52. Petitioner notes that Kohn discloses a dual instruction mode that can issue up to three operations each clock cycle to support parallel execution of functional units. *Id.* (citing Ex. 1008, 16, 24–25). According to Petitioner, the instructions maintained by the sequencer, i.e., the instruction cache, which is coupled to the memories that make up the general purpose register block (the data cache in the i860), include instructions to perform vertex and pixel operations. *Id.* at 52.

5. *Petitioner’s Contentions - Claim 3*

a) *Preamble*

Petitioner cites its discussion at §2[preamble]¹⁴ in support of its contention that the combination of Amanatides and Kohn teach a unified shader. Pet. 52.

b) *Limitations 3[a] and 3[c]*

Petitioner designates claim limitation 3[a] as reciting a processor unit operative to perform vertex and pixel calculation operations. Pet. 30. Petitioner designates claim limitation 3[c] as reciting that the processor unit uses shared resources for either vertex data or pixel information and “is operative to perform pixel calculation operations until enough shared resource become available and then use the shared resource to perform vertex calculation operations.” *Id.* Petitioner addresses these limitations together.

¹⁴ Petitioner discusses the preamble of claim 2 in the Petition at section VII.E.1.a. Pet. 45. We understand Petitioner’s references to §2[preamble], and subsequent references using the § sign to reference corresponding claim limitations, not sections of the Petition.

Petitioner cites Amanatides' use of general purpose i860 processors that perform both floating point and integer operations in parallel, taking advantage of pipeline capability to perform vertex and scalar operations. Pet. 54–55 (citing Ex. 1007, 158; Ex. 1008, 22–23; Ex. 1003, Pfister Decl. ¶¶ 183–185). Petitioner also cites Amanatides' automatic load balancing and priority allocation system as facilitating the performance of more pixel or vertex operations depending upon the vertex/pixel workload, i.e., the amount of vertex or pixel data waiting to be operated on or the amount of memory available for each type of work. *Id.* at 53–54. Petitioner notes that Amanatides' priority system allocates priority to vertex or pixel data so that pixels are processed until none remain or the processor cannot store additional pixels, or so that vertices are processed until none remain or the processor cannot store additional vertices. *Id.* at 53 (citing Ex. 1003, Pfister Decl. ¶ 183; Ex. 1007, 158–159 (mapping GMs to low priority packets and RMs to high priority packets, such that each processor oscillates between working on GMs and RMs)).

Petitioner states that either GMs or RMs could be assigned high priority and that the amount of storage for vertex and pixel data is determined based on the priority given to each type of data. Pet. 55 (citing Ex. 1007, 158; Ex. 1003, Pfister Decl. ¶¶ 184–185). Petitioner also cites Amanatides' disclosure that its bidirectional FIFO chips can be programmed to raise signals at various levels of filling. *Id.* (citing Ex. 1007, 158; Ex. 1003, Pfister Decl. ¶ 185).

Petitioner contends Amanatides discloses “processor units’ that prioritize or load balance either vertex operations or pixel operations, at different degrees of completion, and based on the amount of vertex work or pixel work needs to be performed and/or the amount of memory space or

shared resources available for each type of processing.” Pet. 56 (citing Ex. 1003, Pfister Decl. ¶¶ 184–185). Thus, Petitioner argues, Amanatides and Kohn disclose the processor limitations of claim 3 and the workload limitations of claim 1, 4, 5, and 11, directed to the processor unit, general purpose register block, and sequencer/instruction store. *Id.*

c) Limitation 3[b]

Referencing its arguments concerning limitation 2[a], Petitioner argues that the claimed shared resources can include the general purpose register block, which in Amanatides can encompass the bus, FIFO, DRAM and VRAM for each i860 processor, as well as the instruction and data cache, the memory management unit, the bus control unit, floating point registers, buses, and other circuitry shown in Kohn. Pet. 56. Petitioner notes that these resources are shared for purposes of pixel and vertex processing and that the bus, FIFO, DRAM and VRAM are operatively coupled to the i860 processor. *Id.*

6. Petitioner’s Contentions – Claims 4, 5, 11

The preambles of claims 4, 5, and 11 recite a unified shader. Petitioner cites its discussion of claim limitation 2[a] in support of its contentions that Amanatides and Kohn disclose a unified shader. Pet. 57.

Petitioner cites its discussion of claim limitations 3[a] and 3[c] in support of its contentions that Amanatides and Kohn disclose the processor unit operative to use shared resources to perform vertex calculation operations until sufficient resources become available to perform pixel calculation operations recited in limitations 4[a] and 4[c]. *Id.*

Petitioner cites its discussion of claim limitation 3[b] in support of its contention that Amanatides and Kohn teach the shared resources coupled to the processor unit recited in limitation 4[b].

Petitioner cites its discussion of the preamble of claim 2 and processor unit of limitation 2[b] as support for its contentions that Amanatides and Kohn in combination teach the preamble of claim 5 and processor unit limitation 5[a]. Pet. 57. Petitioner cites its discussion of claim limitations 2[c], [3a], and 3[c] as support for its contention that Amanatides and Kohn in combination teach the sequencer recited in limitation 5[c]. *Id.*

7. *Petitioner's Contentions - Dependent Claim 6*

Claim 6 depends from claim 5 and recites that the sequencer further includes circuitry operative to fetch data from a memory. Ex. 1001, 8:9–10. Petitioner notes that the sequencer in Amanatides includes an on-chip instruction cache and that Kohn describes the instruction cache in the i860 processor is connected to a memory management unit translates virtual address into physical addresses that are issued to an external memory via a 32-bit external address bus. Pet. 58 (citing Ex. 1003, Pfister Decl. ¶ 193, Ex. 1008, 17, 20, Fig. 2). Noting the data from the external memory is received via a 64-bit external data bus near the bus control unit and stored in the data cache for processing, Petitioner asserts that the circuitry of the i860 processor that is disclosed and the combination of Amanatides and Kohn features disclose the limitations of claim 6. *Id.*

8. *Petitioner's Contentions - Dependent Claims 7 and 10*

Claim 7 depends from claim 5 and recites including a selection circuit operative to provide information to the store in response to a control signal. Ex. 1001, 8:11–13. Claim 10 depends from claim 7 and recites that the control signal is provided by an arbiter. Ex. 1001, 8:20–21. Petitioner cites its discussion of claim limitations 3[a] and 3[c] stating that Amanatides alone or in combination with Kohn discloses or renders obvious an arbiter that determines whether to provide a pixel or vertex input to the processor

units as well as the claimed selection circuit and control signals. Pet. 58. According to Petitioner, the “selection circuit” of claim 7 is met by at least the bus and FIFOs that 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 in Amanatides. *Id.* at 58–59. Noting that Amanatides discloses assigning low-priority and high-priority packets to timeslots of the bus (“arbitration”) and that the FIFOs can be programmed to raise signals at various levels of filling, Petitioner asserts that arbitration happens on the message passing bus and would be performed by larger circuits that are not further described in Amanatides, but would be recognized as necessarily present by an ordinarily skilled artisan. *Id.* at 59–60.

9. *Petitioner’s Contentions - Dependent Claims 8 and 9*

Claim 8 depends from claim 5 and recites that the processor unit executes instructions that generate a pixel color in response to the selected data. Ex. 1001, 8:14–16. Claim 9 depends from claim 5 and recites that the processor unit generates vertex position and appearance data in response to selected data. Ex. 1001, 8:17–19.

As to dependent claims 8 and 9, Petitioner cites its discussion of the claimed functionalities in its discussion of claim limitation 2[d]. Pet. 60.

10. *Petitioner’s Contentions - Claim 11*

As to the unified shader in the preamble of claim 11, Petitioner cites its discussion of the preamble of claim 2. Pet. 60. As to the processor unit that performs vertex manipulation operations and pixel manipulation operations based on the vertex or pixel workload and at various degrees of completion based on switching between instructions in the instruction store, Petitioner cites its previous discussion of claim limitations 2[c], 2[d], 3[a],

and 3[c]. *Id.* Citing its discussions of claim limitations 2[c], 2[d], 3[a], and 3[c], Petitioner states that the “instruction store” limitation of claim 11 is met by the same components that satisfy the “sequencer” limitation of claim 2. *Id.* at 61 (citing Ex. 1003, Pfister Decl. ¶ 201).

11. *Petitioner’s Contentions - Claim 1*

Petitioner characterizes independent claim 1 as a method claim that puts the limitations of apparatus claim 3 in method claim format. Pet. 61. According to Petitioner, the steps in claim 1 align with functional elements of the processor unit disclosed by Amanatides, which teaches giving vertex operations priority as long as there is sufficient space in the general-purpose register for vertex processing. *Id.* (citing Ex. 1003, Pfister Decl. ¶ 202). Noting that the “instruction store” limitation is met for the same reasons as the “sequencer” limitation of claim 2, Petitioner contends that claim 1 would have been obvious for the same reasons as claims 2 and 3. *Id.*

12. *Patent Owner’s Contentions and Analysis*

Patent Owner responds to Petitioner’s contentions in this challenge by addressing individual issues common to certain groups of claims, i.e., all claims, claims 1 and 3–11, and claims 2 and 5. Patent Owner does not present particularized arguments concerning any other claims. Recognizing that the burden of persuasion is on the Petitioner, we analyze Petitioner’s contentions in the context of Patent Owner’s responsive arguments below.

a) *All Claims*

As discussed above, Petitioner argues that Amanatides discloses a unified shader. Patent Owner argues that Amanatides does not disclose a unified shader because it does not disclose a shader that can switch back and forth between unfinished vertex and pixel threads. Prelim. Resp. 44. Responding to Dr. Pfister’s testimony concerning the multi-stage structure

of the i860 processor and its dual-operation mode, Patent Owner argues that Petitioner's citations do not discuss performing vertex and pixel operations, let alone doing so simultaneously, or why a floating-point unit would be appropriate for both types of operations. *Id.* Patent Owner also argues that Kohn does not discuss switching between vertex and pixel operations, much less switching between unfinished operations. *Id.* at 45.

Patent Owner's arguments, which are based on its proposed definition of a unified shader, are unavailing, as we do not construe a unified shader to include the features discussed by Patent Owner. *See* Section VIII. B herein. Consistent with the '454 patent specification, we construe a unified shader to mean a shader that is configured to perform both vertex and pixel operations. *See id.*

We also note that, using the i860 processor described in Kohn, Amanatides implements a priority system in which GMs are processed into RMs (pixel data) and RMs are routed from output FIFOs to available input FIFOs until the output FIFOs are empty and space is available in an input FIFO for GMs (vertex) data. Ex. 1007, 159–160. “What will eventually happen is that each processor will oscillate between working on RMs and GMs; in both cases, doing useful work.” *Id.* at 160. As Amanatides discloses that “[w]henver space appears at any input FIFO it is the RMs that are delivered to it” (*id.*), Amanatides does not appear to preclude switching from processing unfinished vertex threads to processing pixel threads.

b) Claims 2 and 5

In response to Petitioner's contentions that Amanatides and Kohn disclose the sequencer recited in limitations 2[c] and 5[b] as discussed above, Patent Owner contends that there is no process in Amanatides or

Kohn for selecting data from a data store; instead, Amanatides simply uses a round robin assignment in which the next data in line is sent to the next processor out of the group. Prelim. Resp. 48 (citing Ex. 1007, 157).

Claim 2 recites the sequencer coupled to the general purpose register block maintains instructions to execute vertex and pixel operations on “selected data maintained in the general purpose register.” Ex. 1001, 7:1–6. Claim 2 does not recite a specific procedure for selecting the data to be maintained in the general purpose register.

Patent Owner contends that “[s]elected data’ is something affirmatively chosen from the data store among options.” Prelim. Resp. 48 (citing Ex. 1001, 2:62–3:3, 5:32–52). The subject matter of the ’454 patent cited by Patent Owner does not support Patent Owner’s contention. At column 2, line 62 through column 3, line 3, referring to Figures 2A and 2B, the ’454 patent states “the shader accepts the vertex coordinate data . . . as inputs and provides data representing, for example, a perspective corrected view” obtained “by applying an appropriate transformation matrix to the data.” At column 5, lines 32–52, the ’454 patent discusses switching between programs or instructions and states that if the general purpose register does not have sufficient space to store incoming vertex data, the vertex data will not be transmitted to the general purpose register and ongoing pixel operations will continue until space becomes available. Neither passage of the ’454 patent cited by Patent Owner describes the claimed selected data as something affirmatively chosen from the data store among options.

Patent Owner also argues that the Amanatides system giving “priority” to certain data over others is not a means by which data is selected from the general purpose register block (referring to claim 2) or maintained

in a store (referring to claim 5). We note, however, that claim 2 recites data “maintained in the general purpose memory block.” Amanatides selects the data to be loaded into the FIFO by giving priority to rendering messages (RMs) and Amanatides allows vertex data (GMs) to access the general purpose bus and be loaded into the FIFO, i.e., the general purpose memory block, only if space is available, a circumstance that occurs after all RMs are processed. Ex. 1007, 159–160. To the extent that “selected data” were interpreted to mean data that is affirmatively chosen, Amanatides’ approach is similar to that described in the ’454 patent in which vertex data is not loaded into the general purpose memory block unless space is available.

Having reviewed the arguments and evidence of record, we are persuaded Petitioner has demonstrated that a person of ordinary skill would have had reason to combine the teachings of Amanatides and Kohn, and that in combination, Amanatides and Kohn would have taught or suggested the limitations of claim 2 and claim 5 to an ordinarily skilled artisan.

c) Claims 1, 3–11

As discussed above, Petitioner contends that the priority and load balancing features of Amanatides and Kohn disclose limitations that recite continuing to perform vertex or pixel operations until resources, e.g., registers in the general purpose register, become available or have sufficient space to store the type of data not being processed. Patent Owner contends that claims 1 and 3–11 all require performing operations based on dynamically monitored data storage. Prelim. Resp. 15–16. Patent Owner does not explain the meaning of “dynamically monitored.” *See id.*

Patent Owner contends that “neither Amanatides nor Kohn discusses considering the amount of space available in the store, let alone that consideration impacting processing of particular data.” Prelim. Resp. 46.

According to Patent Owner, “the sole consideration in Amanatides for load balancing is the amount of a given type of work, such as texture mapping, that needs to be performed,” but “[t]here is no consideration of data storage capacity.” *Id.* (citing Ex. 2001, Mangione-Smith Decl. ¶ 326). Patent Owner further argues that Amanatides’ priority scheme, i.e., mapping GMs to low priority packets and RMs to high priority packets and denying access to the data bus unless there are no RMs waiting to be sent, is a system that determines what to transmit solely based on data type, rather than current data storage space. *Id.* at 47.

Although Patent Owner is correct that Amanatides’ priority system delivers RMs before GMs, Patent Owner’s arguments are unavailing. Amanatides’ priority system explicitly provides for delivering vertex data to the input the FIFOs, i.e., the claimed registers in the general register (claim 1), the claimed shared resources (claims 3, 4), or the claimed store (claims 5–10), when sufficient space becomes available to store it. Ex. 1007, 160. Amanatides notes that “[t]he worst possible scenario is when every input queue is full of GMs.” *Id.* After processing, GMs that are converted to RMs, the RMs are delivered to any input FIFO that has space for it. *Id.* “[W]hen all the output FIFOs are empty and the processors are working on the remaining RMs in their input FIFOs, the processors can begin to deliver GMs again.” *Id.* As Amanatides delivers pixel and vertex data to FIFOs based on the available space, we agree with Petitioner that Amanatides discloses this limitation.

The basis for Patent Owner’s assertion that claim 11 requires monitoring the amount of data storage is unclear. Patent Owner cites the limitation that the processor unit performs vertex and pixel manipulation operations at various degrees of completion based on switching between

instructions in the instruction store, but does not explain how this limitation implicates monitoring the amount of data storage. Prelim. Resp. 16.

Having reviewed the arguments and evidence of record, we are persuaded Petitioner has demonstrated that a person of ordinary skill would have had reason to combine the teachings of Amanatides and Kohn, and that in combination, Amanatides and Kohn would have taught or suggested these limitations of claims 1 and 3–11 to an ordinarily skilled artisan.

13. *Remaining Issues*

We have considered Petitioner’s other arguments and evidence particularized to each of claims 1–11 under this ground. After reviewing the arguments and evidence of record, we are persuaded Petitioner has demonstrated that a person of ordinary skill would have had reason to combine the teachings of Amanatides and Kohn, and that in combination Amanatides and Kohn would have taught or suggested the limitations of claims 1–11 to an ordinarily skilled artisan.

14. *Conclusion*

Having considered the arguments and evidence of record, we determine that Petitioner has demonstrated a reasonable likelihood that it will succeed in its challenge to claims 1–11 as obvious over the combination of Amanatides and Kohn.

D. Petitioner’s Challenge to Claims 1–11 As Obvious Over Selzer and Fiske

1. *Selzer (Ex. 1009)*

Selzer discloses a graphics processor with an arbitrary number of high-level programmable processors able to perform both rendering tasks and geometry calculations operating in parallel. Ex. 1009, 37, 39. An example digital signal processor (DSP) has an on-chip cache, fast instruction

cycle, floating point unit, and two independent parallel bus interfaces. *Id.* at 40. Figure 4.4 of Selzer, reproduced below, is a block diagram of Selzer's system.

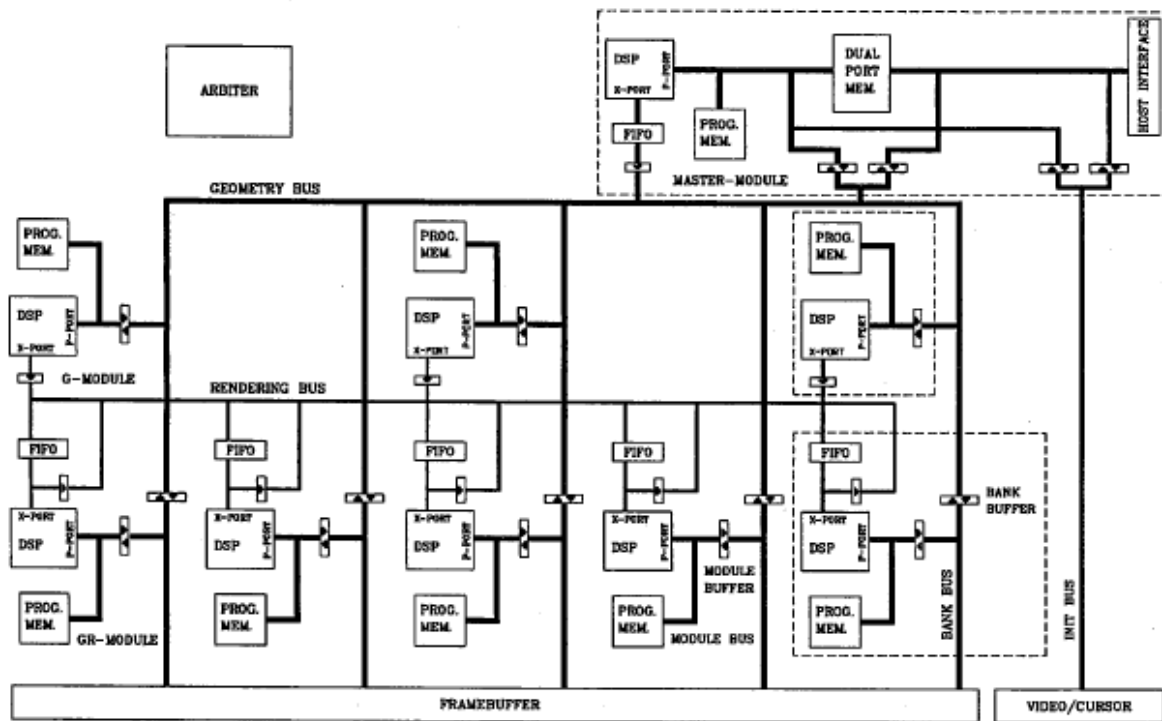


FIGURE 4.4. System architecture

Id. at Fig. 4.4. Architecturally, Selzer's system is arranged as a master module, a geometry module, and a rendering module. The system is organized as a pipeline with additional parallelism on a functional level using multiple geometry and rendering modules. *Id.* at 43, Fig. 4.4 (showing a master module, three geometry modules (G), and five rendering modules (G-R)). All modules are connected to a geometry bus that acts as a system bus for transferring system, graphics, and update data. *Id.* A rendering bus is designed to convey only rendering primitives to data FIFOs in rendering modules. *Id.*

The master module communicates with a host processor via a dual port memory, allowing data to be transferred and processed in parallel.

Ex. 1009, 41. All system resources are accessible by the master. *Id.* at 43. The master module traverses the graphics data structure and feeds graphics data, e.g., high order primitives like splines, polygons, meshes, or triangles, into the data FIFO for delivery to appropriate processors. *Id.* If a geometry module has finished the last task, it accesses the geometry bus and fetches the next primitive or task automatically. *Id.*

A graphics module performs transformation, clipping, polygon and patch subdivision, normal interpolation, and renormalization and lighting operations; the graphics module delivers the processed graphic primitives to the rendering module data FIFOs. *Id.* A rendering module fetches data from its data FIFO, performs rendering calculations, e.g., shading and scan conversion, and conveys the calculated pixel values to the frame buffer. *Id.*

The architecture achieves peak performance when all rendering processor power is exploited for rendering calculations and the rendering processors are working continuously. Ex. 1009, 48. Recognizing that frequently used features, e.g., small triangles or short vectors parallel to the x and y axis, can be rendered quickly, but that geometric calculations for interpolating shading methods cannot be accelerated because they are independent of the resulting size of the primitive, Selzer's system uses "dynamic load balancing" or "task switching" to speed up geometry calculations. *Id.* at 45–46. The structure of Selzer's rendering module is similar to that of a geometry module, allowing a rendering module to act as a geometry module by fetching graphics data from the master data FIFO and delivering processed data to the FIFOs of an appropriate rendering module. *Id.* After rendering an object, if there is no rendering data in a rendering processor's input buffer, rather than remain idle, the processor switches to

geometry processing by requesting an unprocessed geometry object and continuing geometry calculations. *Id.* at 47.

2. *Fiske (Ex. 1010)*

Fiske discloses a parallel processor system using multiple context processors that provide multiple register sets, including multiple instruction pointers, that allow multiple threads to be loaded and ready to run at the same time. Ex. 1010, 210. When a currently executing thread misses in the cache or fails a synchronization test, the processor can begin executing one of the other threads loaded in one of the other hardware contexts. *Id.* Fiske refers to a “context switch” as occurring when the processor switches from one loaded thread to executing another loaded thread, and a “thread swap” as involving swapping a loaded thread with an unloaded thread from a software queue. *Id.*

Figure 1 of Fiske, reproduced below, illustrates a multiple context processor with N contexts.

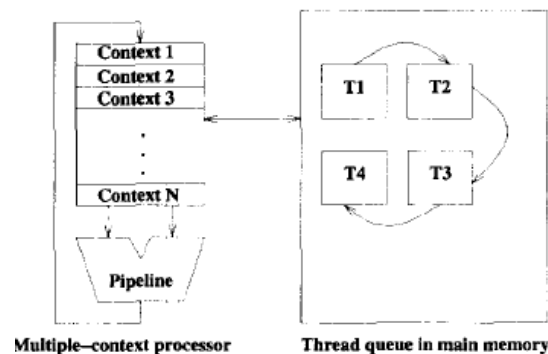


Figure 1: Multiple-context processor with N contexts.

Ex. 1010, Fig. 1. Fiske discloses allowing rapid context switching between several threads as a means of tolerating communication and synchronization latencies. *Id.* at 210. Fiske addresses scheduling such threads, i.e., first determining which threads should have their state loaded in the multiple

contexts, and second, deciding which loaded thread to execute at any given time. *Id.* Fiske recognizes that if critical threads are loaded along with non-critical threads and the scheduler treats all loaded threads equally, runtime performance can suffer. *Id.* at 217. Fiske addresses this issue with thread prioritization in which each thread's priority indicates the importance of the thread to the overall problem and changes dynamically. *Id.* at 216–217. First, at each node a software scheduler chooses the highest priority threads as the loaded threads, with threads of equal priority scheduled in a round-robin fashion. *Id.* at 217. Second, at a context switch, the hardware scheduler chooses the loaded thread with the highest priority as the next thread; but if a thread is waiting for a memory reference to be satisfied, it is stalled and is not considered for scheduling until the memory reference is satisfied. *Id.*

3. *Reasons to Combine*

Petitioner contends that a person of ordinary skill would have had reason to combine the flexible graphics processing teachings of Selzer and the multithreaded architecture of Fiske because they both address load balancing within a processing system. Pet. 64. Petitioner notes that Fiske incorporates multiple hardware thread contexts to allow for rapid switching among threads, Selzer utilizes a scalable approach to parallelism by multiplying geometry and rendering modules. *Id.* (citing Ex. 1003, Pfister Decl. ¶¶ 221–226). Noting that Selzer explains how three geometry modules and five rendering modules can be used to switch from performing pixel tasks to performing vertex tasks, Petitioner states that Fiske condenses the approach into a single processing module that itself includes multiple thread contexts. *Id.* at 64–65. According to Petitioner, a person of ordinary skill would have recognized that implementing Fiske's multithreaded

processing architecture as the rendering modules of Selzer would have resulted in performing the same operations as Selzer's rendering models but would save chip space. *Id.* at 65 (citing Ex. 1003, Pfister Decl. ¶ 223).

Petitioner also notes that an ordinarily skilled artisan would recognize that both Selzer and Fiske are directed to dynamic load balancing and the scheduling approach of either reference could be utilized in their combination. *Id.* at 65.

Patent Owner does not argue explicitly that a person of ordinary skill would not have had reason to combine the teachings of Selzer and Fiske. Based on the arguments and evidence currently of record, we are persuaded that Petitioner has demonstrated a person of ordinary skill would have had reason to combine the teachings of Selzer and Fiske.

4. *Petitioner's Contentions - Claim 2*

a) *Claim 2 Preamble*

Petitioner cites Selzer as disclosing a shader architecture with a load balancing mechanism that distributes processing between a geometry section and a rendering section. Pet. 66–67 (citing Ex. 1009, 37). Petitioner argues that, because Selzer's rendering module performs both vertex and pixel operations, Selzer teaches a unified shader, as defined in the '454 patent. *Id.* at 67 (citing Ex. 1009, 40–41; Ex. 1003, Pfister Decl. ¶ 227). Petitioner further argues that, to the extent the unified shader is unnecessarily read to require a multithreaded processor unit, combination of Selzer and Fiske, with its multiple thread contexts, teach this feature. *Id.* (citing Ex. 1003, Pfister Decl. ¶ 228).

a) *Limitation 2 [a]*

As to the general purpose register block recited in limitation 2[a], Petitioner cites Selzer as disclosing each of its rendering processors and the

master processor include a block of registers in the form of programmable memories and FIFOs that receive and maintain all geometry (vertex) and rendering (pixel) data. Pet. 68 (citing Ex. 1009, 44–45; Ex. 1003, Pfister Decl. ¶ 229). Petitioner notes that Selzer’s master module directs either type of graphics data to a rendering module over the system bus in order to achieve load balancing. *Id.* at 69 (citing Ex. 1009, 41, 43–44; Ex. 1003, Pfister Decl. ¶ 229). Petitioner further notes that Fiske teaches loaded thread contexts that include data and instructions for execution that can be selected by a thread prioritization scheduling scheme and sent to the processor unit. *Id.* at 69–70 (citing Ex. 1003, Pfister Decl. ¶ 230). Thus, according to Petitioner, Selzer alone, or the combined teachings of Selzer and Fiske, disclose the general register block recited in limitation 2[a].

b) Limitations 2[b] and 2[d]

Petitioner asserts that Selzer alone or in combination with Fiske teaches the claimed processor unit (limitation 2[b]) that executes instructions to generate a pixel color in response to selected data from the general purpose register block and to generate vertex position and appearance data in response to selected data from the general purpose register block (limitation 2[d]). Pet. 70–71. Petitioner notes that Selzer includes a rendering module that may perform vertex operations, i.e., generating vertex position and appearance data, which Selzer describes as transformation, clipping polygon and path subdivision, normal interpolation, and lighting operations. *Id.* (citing Ex. 1009, 42; Ex. 1003, Pfister Decl. ¶ 234). Petitioner also notes that Selzer’s rendering module generates pixel color and conveys the calculated pixel colors and values to the frame buffer. *Id.* at 71. Finally, Petitioner cites Selzer as disclosing vertex and pixel operations are based on

respective vertex and pixel data and instructions selected from the registers under coordination of the master module. *Id.*

c) Limitation 2[c]

Petitioner cites Selzer alone or in combination with Fiske as teaching the sequencer recited in claim limitation 2[c]. Pet. 71. The sequencer limitation of claim 2[c] recites that the sequencer is coupled to the general purpose register block and the processor unit and maintains instructions that cause the processor unit to execute vertex calculations and pixel calculations on selected data maintained in the general purpose register block. Ex. 1001, 7:1–6. Petitioner points out that Selzer’s master module accesses graphics data in the system and fetches the next primitive or task automatically. Pet. 71 (citing Ex. 1009, 43). Noting that Selzer’s master module gathers graphics data and tasks and supplies those tasks in the form of data and instructions to the appropriate processor unit, Petitioner contends that the master module is able to switch between sending vertex or pixel tasks to each rendering unit, allowing the rendering units to load balance among vertex and pixel graphics operations. *Id.* at 70–71 (citing Ex. 1009, 40–41; Ex. 1003, Pfister Decl. ¶236). Noting that Selzer alone discloses this limitation, Petitioner also cites Fiske as providing more detail in the context of a multithreaded processing unit. *Id.* (citing Ex. 1003, Pfister Decl. ¶237). According to Petitioner, when the multiple thread contexts of Fiske are present in a rendering module of Selzer, Fiske’s scheduling algorithm chooses the active thread for which the rendering module would execute the next instruction, which instruction could be for a vertex calculation or a pixel calculation. *Id.* at 72 (citing Ex. 1003, Pfister Decl. ¶238). Noting that Fiske assigns a priority to each and every thread in the system, Petitioner states that the instruction selection can move back and forth

between vertex and pixel operations very rapidly. *Id.* (citing Ex. 1003, Pfister Decl. ¶ 238).

5. *Petitioner's Contentions - Claim 3*

a) *Claim 3 Preamble*

As to the preamble of claim 3, Petitioner cites its discussion of the preamble of claim 2. Pet. 72–73.

b) *Limitations 3[a] and 3[c]*

Claim limitations 3[a] and 3[c] recite a processor unit operative to perform vertex and pixel calculations and to use shared resources for either vertex data or pixel information, perform pixel calculation operations until enough shared resources become available, and then use shared resources to perform vertex calculation operations. Ex. 1001, 7:13–14, 7:16–21.

According to Petitioner, Selzer's system is dynamic and can adjust to process either vertex or pixel operations in its rendering module. Pet. 73 (citing Ex. 1003, Pfister Decl. ¶ 241). Petitioner cites an example in Selzer describing how a rendering model prioritizes pixel operations, but if it reaches an idle state, when no pixel resources are available, the rendering module switches to prioritize and perform vertex operations. *Id.* (citing Ex. 1009, 47; Ex. 1003, Pfister Decl. ¶ 241). Petitioner contends that an ordinarily skilled artisan would have understood that this automatic load balancing architecture in Selzer prioritizes either vertex or pixel operations depending on the processing load present in the system. *Id.*

Although Petitioner contends that Selzer alone discloses limitations 3[a] and 3[c], Petitioner further cites Fiske, with its multiple thread contexts, as scheduling vertex or pixel operations to achieve load balancing. Pet. 73. According to Petitioner, in Fiske, if shared resources are not available for pixel processing, the Selzer/Fiske combination would allocate its thread

contexts for vertex processing vice versa. *Id.* at 73–74 (citing Ex. 1010, 210–211; Ex. 1003, Pfister Decl. ¶ 242). Petitioner notes that once thread contexts are allocated, Fiske performs another selection to determine which resources should be used for vertex or pixel operations. *Id.* According to Petitioner, “Fiske describes this process being done in a fine-grained manner that can switch among different instructions of a thread, and at a time when the entire thread is not completed.” *Id.* at 74.

Petitioner further argues that the multithreaded processor of Fiske performs vertex or pixel operations based on the vertex or pixel workload and can switch between instructions in the instruction store to perform vertex and pixel operations at various degrees of completion, i.e., Fiske can switch between unfinished vertex threads and unfinished pixel threads. Pet. 74 (citing Ex 1003, Pfister Decl. ¶ 242). Thus, according to Petitioner, the combination of Selzer and Fiske works in a multithreaded environment that meets not only these limitations of claim 3 but also the workload-related limitations of claims 1, 4, 5, and 11. *Id.*

c) Limitation 3[b]

As to the shared resources recited in claim limitation 3[b], Petitioner cites its discussion of claim limitation 2[a]. Pet. 74. Petitioner’s discussion of limitation 2[a] identifies Selzer’s master register blocks and Fiske’s multiple thread contexts as disclosing this limitation.

6. Petitioner’s Contentions - claims 4, 5, 8, 9, 11

As to the preamble of independent claim 5 and claim limitation 5[a], Petitioner cites its discussion of the preamble of claim 2 and claim limitation 2[b] of this ground; as to claim limitation 5[b], Petitioner cites its discussion of limitations 2[c], 3[a], and 3[c] and Dr. Pfister’s discussion of the scheduling architecture in Fiske as able to dynamically adjust to select any

type of data based on resources. Pet. 75 (citing Ex. 1003, Pfister Decl. ¶ 248).

As to claim 8, which depends from claim 5, and recites the processor unit executes instructions that generate a pixel color in response to the selected data, Petitioner cites its discussion of claim limitation 2[d] in this ground. Pet. 77.

As to claim 9, which depends from claim 5 and recites the processor generates vertex position and appearance data in response to a selected data, Petitioner cites its discussion of claim limitation 2[d] in this ground. Pet. 78.

As to the preamble of independent claim 11, Petitioner cites its discussion of the preamble of claim 2 under this ground. Pet. 78. As to the processor unit recited in claim limitations 11[a] and 11[c], Petitioner cites its discussion of claim limitations 2[c], 2[d], 3[a], and 3[c]. *Id.* Petitioner also cites Dr. Pfister’s explanation of how the scheduler in Fiske can switch between threads at various levels of completion in order to dynamically load balance. *Id.* (citing Ex. 1003, Pfister Decl. ¶ 256).

7. *Patent Owner’s Contentions and Analysis*

Patent Owner responds to Petitioner’s contentions in this challenge by addressing individual issues common to certain groups of claims, i.e., all claims, claims 1 and 3–11, and claims 2 and 5. Patent Owner does not present particularized arguments concerning any other claims. Recognizing that the burden of persuasion is on the Petitioner, we analyze Petitioner’s contentions in the context of Patent Owner’s responsive arguments below.

a) *All Claims*

Patent Owner contends that the combined teachings of Selzer and Fiske do not disclose a unified shader, because “the purported processor cannot switch between unfinished threads at all, be they pixel, vertex, or

other.” Prelim. Resp. 49 (citing Ex. 2001, Mangione-Smith Decl. ¶ 333). According to Patent Owner, the entire architecture of Selzer depends on the rendering unit fully completing its assigned task, i.e., Selzer does not monitor resource availability, but simply states that if no more pixel work remains to be done, the rendering module can do vertex work, instead, or go idle. *Id.* at 49–50 (citing Ex. 2001, Mangione-Smith Decl. ¶¶ 334–335). Patent Owner also contends that the Petition fails to explain or provide evidence as to how Fiske’s thread prioritization algorithm, which guides which thread has its instructions executed at a node, teaches switching between unfinished threads. *Id.* at 50–51.

Patent Owner’s arguments, which are based on its definition of a unified shader, are unavailing, as we do not construe unified shader to include the features discussed by Patent Owner, i.e., the ability to switch back and forth between executing *unfinished* vertex and pixel threads. *See* Section VIII. B herein. Consistent with the ’454 patent specification, we construe a unified shader to mean a shader that is configured to perform both vertex and pixel operations. *See id.* As discussed above, Patent Owner acknowledges that Selzer discloses the rendering module can perform vertex and pixel processing. Prelim Resp. 49–51. Thus, we are persuaded that Selzer and Fiske disclose a unified shader, as recited in all the claims.

b) Claims 2 and 5

Patent Owner contends Petitioner has not demonstrated that the combined teachings of Selzer and Fiske disclose or suggest a shader that executes pixel or vertex thread instruction “in response to” receiving selected data from a data storage (i.e., claim limitation 2[d]). Prelim. Resp. 53. Citing Petitioner’s discussion of the thread prioritization scheduling scheme in Fiske, Patent Owner argues that, in Fiske, receipt and loading of

threads, or groups of instructions, causes execution, not receipt, of data. *Id.* (citing Ex. 1010, 211 (for the proposition that a software scheduler uses priority to decide which threads to load and a hardware scheduler uses priority to determine which loaded thread to execute); Ex. 2001, Mangione-Smith Decl. ¶ 341).

Selzer discloses a master module that traverses the graphics data structure and feeds graphics data to a FIFO for delivery to appropriate processors. Ex. 1009, 41. The geometry module or rendering module then processes the data. Selzer discloses that to exploit the distributed computational power of the system, processing units must adapt their activities to the actual processing requirements of the scene. *Id.* at 46. Selzer accomplishes this goal with processors that switch from processing pixel data to geometry calculations to avoid entering an idle state. *Id.* at 46–47. To perform such task switching, the processor accesses the geometry bus and requests new unprocessed geometry objects. *Id.* at 43, 47. As Selzer discloses performing rendering or geometry processing based on the data received, we determine that, for purposes of institution, Petitioner has demonstrated Selzer discloses the subject claim limitations.

c) Claims 1 and 3–11

Noting that each of claims 1 and 3–11 requires considering “the amount of space for storage impacting processing or data selection,” Patent Owner contends that “neither Selzer nor Fiske discusses considering the amount of space available in the store, let alone considering impacting the processing of particular data.” Prelim. Resp. 52. According to Patent Owner, assigning priorities to threads, data, or data types does not disclose determining what type of data to transmit and process based on an evaluation of the current data storage capacity. *Id.* (citing Ex. 1009, 46–47

(discussing the rendering module performing geometry processing after all required rendering processing); Ex. 1010, 211 (stating that thread prioritization involves assigning a priority to all the different threads in an application and using the priority to make thread scheduling decisions to determine which loaded thread to execute); Ex. 2001, Mangione-Smith Decl. ¶ 339).

As discussed above, Petitioner cites Selzer's load balancing techniques as teaching this claim limitation. Selzer's rendering modules are capable of performing both geometry and rendering calculations. Ex. 1009, 46–47. Recognizing that the rendering modules generally perform rendering operations faster than geometry operations, Selzer attempts to balance the processing load. *Id.* at 45–46. After rendering an object, any rendering module that does not have rendering data in its input buffer will switch tasks to request new unprocessed geometry objects and continue geometry calculations. *Id.* at 47. We agree with Patent Owner that, on the current record, Petitioner has not demonstrated how these load balancing teachings of Selzer alone would have disclosed to a person of ordinary skill the claim limitations that recite continuing pixel processing until shared resources, e.g., enough registers in the general purpose register block, become available, and vice versa. Instead, Selzer appears to disclose switching to geometry processing after the input buffer is empty.

As discussed above, Petitioner also cites Fiske's disclosure of multiple context processing and thread prioritization. Pet. 73–74 (citing Ex. 1010, 210–211). Although Fiske may disclose switching between unfinished threads, we agree with Patent Owner that, on the present record, Petitioner does not demonstrate such switching takes place based on the amount of space available in the shared resources. For example, Fiske discloses that if

a thread is waiting for a memory reference to be satisfied, it is stalled and is not considered for scheduling until the memory reference is satisfied.

Ex. 1010, 211.

Having reviewed the arguments and evidence of record, we determine that Petitioner has not demonstrated the combined teachings of Selzer and Fiske would have disclosed or suggested the subject limitations of claims 1 and 3–11.

8. *Conclusion*

Based on the arguments and evidence currently of record, we determine that Petitioner has demonstrated a reasonable likelihood it will succeed on its challenge to claim 2 as obvious over combined teachings of Selzer and Fiske. We are not persuaded that Petitioner has demonstrated the likelihood of success on its challenge to claims 1 and 3–11 on this ground.

X. CONCLUSION

For the reasons discussed above, we determine that Petitioner has demonstrated a reasonable likelihood that it will succeed on the following challenges to patentability:

Claims 1–11 as obvious over Amanatides and Kohn and claim 2 as obvious over Selzer and Fiske.

XI. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that, pursuant to 35 U.S.C. § 314(a) an *inter partes* review of the '454 patent is hereby instituted, commencing on the entry date of this Order, and pursuant to 35 U.S.C. § 314(c) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial;

FURTHER ORDERED that the trial is authorized on all grounds set forth in the Petition; and

FURTHER ORDERED that the trial will be conducted in accordance with a corresponding separately issued Scheduling Order.

IPR2023-00922
Patent 8,760,454 B2

FOR PETITIONER:

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

BACK-UP 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

FOR PATENT OWNER:

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,

IPR2023-00922
Patent 8,760,454 B2

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