Trials@uspto.gov
571-272-7822

Paper 7
Entered: October 25, 2023

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,

v.

ATI TECHNOLOGIES ULC,
Patent Owner.

IPR2023-00564
Patent 7,742,053 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 1–9 (the "challenged claims") of U.S. Patent No. 7,742,053 B2 ("the '053 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 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 Petitioner Realtek Semiconductor Corp. as the real party-in-interest. Pet. 1. Patent Owner identifies ATI Technologies ULC as the real party-in-interest. Paper 4, 1.

III. RELATED MATTERS

Petitioner and Patent Owner identify the following as proceedings that may affect or may 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); and

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

Pet. 1–2; Paper 4, 1.

IV. THE '053 PATENT

The '053 patent relates to graphic processing and the interleaving of arithmetic logic unit (ALU) operations with texture fetching operations. Ex. 1001, 1:16 – 18. According to the '053 patent, in a typical graphics processing system, the processing elements, such as vertices and/or pixels are processed through multiple steps that provide for the application of textures and other processing instructions, as done through one or more ALUs. *Id.* at 1:24 – 27. The '053 patent describes its Figure 1, reproduced below, as "a prior art sequencing system." *Id*. at 1:32.

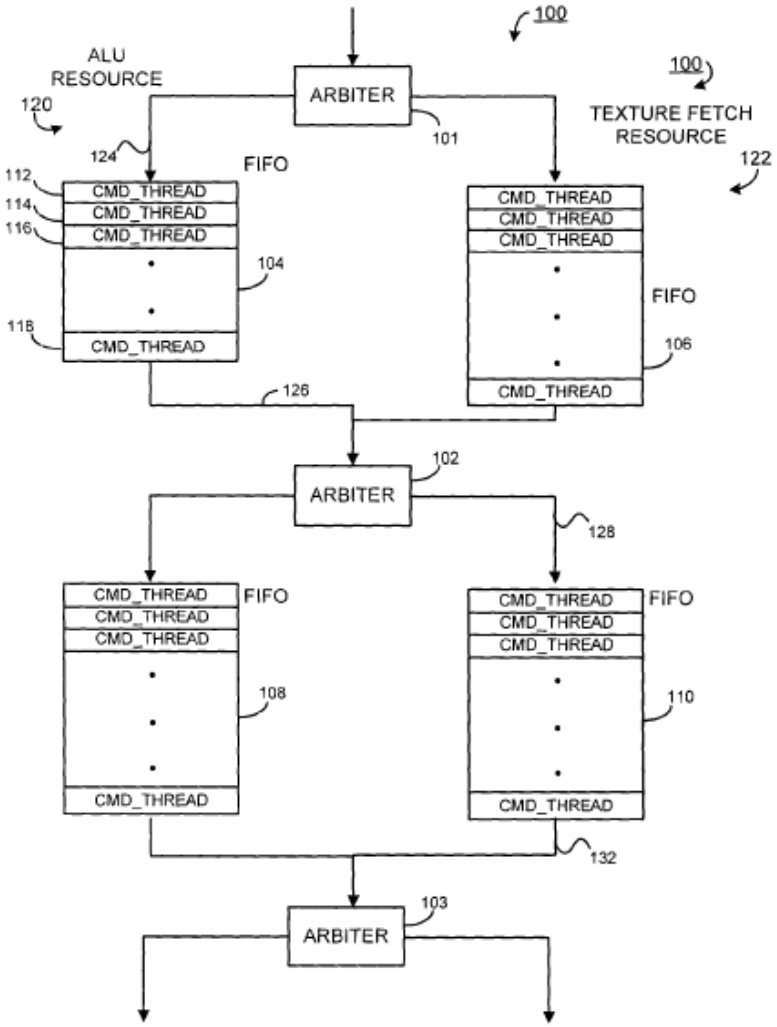


FIG. 1
(PRIOR ART)

Id. at Fig. 1. In Figure 1, system 100 includes first, second and third arbiters 101–103 and multiple buffers or "reservation stations" 104, 106, 108, 110 that are typically first-in, first-out (FIFO) buffers. *Id.* at 1:33–35, 1:43–44, 1:46–47. Each buffer or reservation station stores multiple command threads, e.g., 112, 114, 116, 118. *Id.* at 1:36 – 38; *see also id.* at 2:37–41. "[A] command thread is a sequence of commands applicable to a corresponding element such as a pixel command thread relative to processing of pixel elements and a vertex command thread relative to vertex processing commands." *Id.* at 2:41–45.

Prior art system 100 is divided into ALU resource division 120 and texture fetch resource division 122. Ex. 1001, 1:38–40. In ALU resource division 120, command thread 118 selected by arbiter 101 may be received in a reservation station 104, 108 from an input command 124; command thread 118 may then be withdrawn from reservation stations 104 and 108 and provided to an ALU (not shown). *Id.* at 1:40–45. Command threads within texture fetch resource division 122 may be withdrawn from reservation stations 106 and 110 to be provided to a texture fetch processor (not shown). *Id.* at 1:45–48. First buffer 104 receives input command 124 and outputs a completed command thread 126 to second arbiter 102. *Id.* at 1:49–51. Arbiter 102 receives input command 124 and in due course provides the command thread to either an appropriate texture fetch buffer 110 or an ALU buffer 108. *Id.* at 1:55–57. The steps are repeated where an output thread command 128 is provided to another ALU (not shown) or texture fetch processor (not shown) and returned to buffer 108 or 110. *Id.* at 1:57–61. Buffer 110 also produces output 132 which is a command thread that may be provided to another arbiter 103 to be provided further along the graphics processing pipeline. *Id.* at 1:61–64.

According to the '053 patent, the prior art system in Figure 1 is inflexible because delineated ALU resource buffers and texture fetch resource buffers are such that command threads must be sequentially provided through the various buffers 104, 106, 108, and 110; the system also does not support an unlimited number of dependent fetches based on the buffer structure and their connectivity between each other and the available ALU resources and texture fetch resources. Ex. 1001, 1:65–2:6.

Figure 2 of the '053 patent, reproduced below, is a schematic block diagram of a multithread processing graphics system according to one embodiment of the invention. Ex. 1001, 2:15–17.

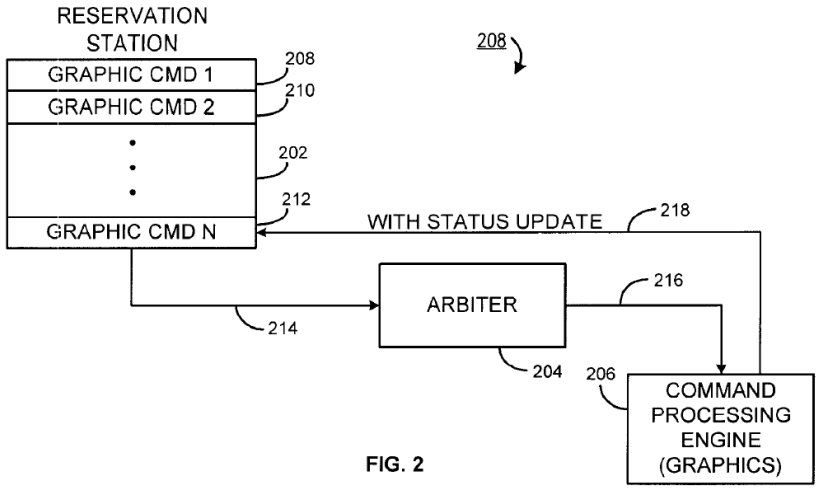


FIG. 2

The embodiment in Figure 2 includes reservation station 202, which can be "any type of memory device capable of reserving and storing command threads," and arbiter 204, implemented in hardware, software or a combination thereof; the arbiter receives a command thread 208, 210, 212 from the reservation station and provides the command thread to command processing engine 206. Ex. 1001, 2:15–17, 2:36–52, 2:63–3:11. Command processing engine 206 performs a thread command and provides status update 218 to the corresponding command thread 208, 210, or 212 in reservation station 202. *Id.* at 3:12–15. The '053 patent provides for processing of multiple threads to prevent the corresponding resources from going idle while waiting for resources to become available, e.g., while awaiting retrieval of specific data. *Id.* at 3:16–20. In one embodiment, a done flag indicates when all commands within a thread have been executed, allowing the command thread to be retrieved from the reservation station and provided to a further processing element within a graphics processing

pipeline. *Id.* at 3:20–28. Arbiter 204 may also receive command threads 208–212 based on a priority scheme appropriate for effective utilization of the arbiter and command processing engine 206, or based on the oldest available thread. *Id.* at 3:29–36.

Figure 3, reproduced below, is a schematic block diagram of a pipeline vector machine according to one embodiment of the invention. Ex. 1001, 2:15–17, 3:29–30.

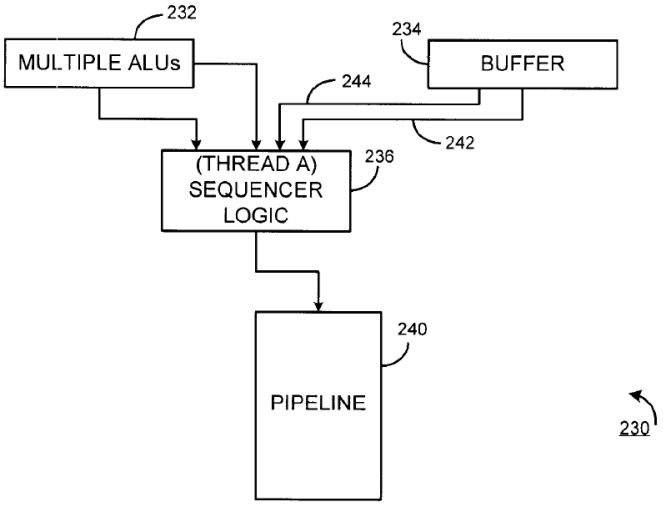


FIG. 3

Id. at Fig. 3. The system in Figure 3 includes multiple ALU system 232, buffer 234, and sequencer logic 236, which may be an ALU resource. *Id.* at 3:37–40. Sequencer logic 236, coupled to pipeline 240, receives first thread 242 from buffer 234, such that logic 236 may perform simultaneous interleaved execution of the command threads. *Id.* at 3:41–45. Like fetch arbitration, ALU arbitration proceeds with the arbiter selecting the first command ready to execute from vertex and pixel reservation stations. *Id.* at 3:48–53.

Figure 4 of the '053 patent reproduced below, illustrates another embodiment of a multithread command processing system. Ex. 1001, 3:62–63.

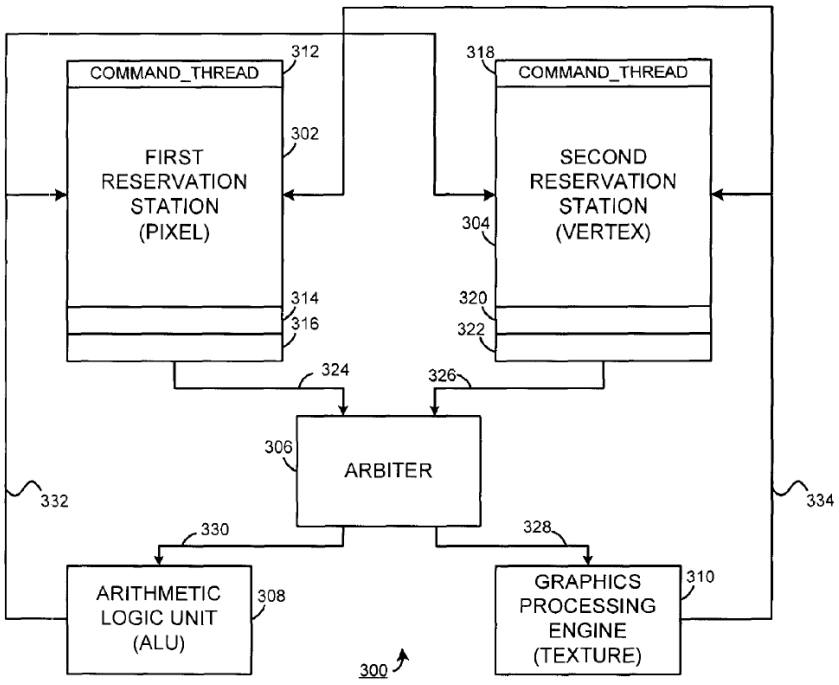


FIG. 4

Id. at Fig. 4. The embodiment in Figure 4 shows pixel reservation station 302 with command threads 312, 314, 316 having pixel based commands and vertex reservation station 304 with vertex command threads 318, 320, 322. *Id.* at 3:63–4:4. Arbiter 306 selectively retrieves either pixel command thread 324 or vertex command thread 326 and provides one thread 328, which may be either pixel thread 324 or vertex thread 326, to graphics processing engine 310, such as a texture engine, and provides the other thread 330 to ALU 308. *Id.* at 4:28–33. Two sets of arbitration are performed, one for pixels, such as command thread 316, and one for vertices, such as command thread 322. *Id.* at 5:13–16. Arbiter 306 selects the proper allocation of which command thread goes to graphics processing engine 310 and which command thread goes to ALU 308 based on state bits

and status bits that indicate the progress of the thread executions. *Id.* at 4:49–5:10. Upon execution of the thread, ALU 308 and graphics processing engine 310 return command threads 332, 334 to the originating reservation stations. *Id.* at 4:34–41. To switch a command thread from ALU 308 to graphics processor engine (texture) 310, that thread must be returned to its reservation station, re-retrieved by arbiter 306 and provided to the other unit 308, 310 respectively. Upon execution of all embedded commands in a thread, as indicated for example by a done flag, ALU 308 provides the completed command thread further in the processing pipeline, e.g., to a suitable rending backend and scan converter, as would be recognized by an ordinarily skilled artisan. *Id.* at 5:19–42, Fig. 5.

V. ILLUSTRATIVE CLAIM

Claims 1 and 5 are independent. Claim 5, reproduced below, is illustrative of the subject matter of the '053 patent:

> 5. A graphics processing system comprising:
>
> at least one *memory device* comprising a first portion operative to store a plurality of pixel command threads and a second portion operative to store a plurality of vertex command threads;
>
> an arbiter, coupled to the at least one memory device, operable to select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads; and
>
> a plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread.

Ex. 1001, 8:4–15.

Claim 6 recites that the plurality of command processing engines comprise at least one arithmetic logic unit; claim 7 recites that the command

processing engines comprise least one texture processing unit. Ex. 1001, 8:16–21.

Independent claim 1 differs from independent claim 5 by (a) reciting that the arbiter selects a command thread based on relative priorities and (b) not reciting a plurality of command processing engines coupled to the arbiter as recited in claim 5. *Compare* Ex. 1001, 7:10–21, 8:4–15. Claim 2 depends from claim 1 and recites a command processing engine coupled to the arbiter, with the arbiter operable to provide the command thread to the command processing engine. *Id.* at 7:22–26. Claims 3 and 8 depend from claim 1 and 5, respectively, and recite the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device. *Id.* at 7:27–30, 8:23–25. Claims 4 and 9 depend from claims 1 and 5, respectively, and recite a single memory device comprises the first portion and the second portion of the at least one memory device. *Id.* at 8:1–3, 8:26–28.

VI. ASSERTED GROUNDS

Petitioner asserts that claims 1–9 would have been unpatentable on the following grounds:

Claim(s) Challenged	35 U.S.C. §	Reference(s)
1–9	103[1]	Stuttard[2], Admitted Prior Art

[1] *See* The Leahy-Smith America Invents Act ("AIA"), Pub. L. No. 112-29, 125 Stat. 284, 287–88 (2011), amended 35 U.S.C. §§ 102, 103, and 112. Because the application of the '053 patent was filed before March 16, 2013 (the effective date of the relevant amendments), the pre-AIA versions of §§ 102, 103, and 112 apply.

[2] PCT Pub. No. WO00/62182A2, pub. Oct. 19, 2000 to Stuttard et al. (Ex. 1005)

Claim(s) Challenged	35 U.S.C. §	Reference(s)
1–9	103	Stuttard, Williams[3]
1–9	103	Williams, Whittaker[4]

The following evidence is also in the record before us:

Declaration of Dr. Massound Pedram In Support Of Petition For *Inter Partes Review* (Ex. 1004, "Pedram Decl.");

Declaration of Dr. John Hart Regarding U.S. Patent 7,742,053 (Ex. 2001, "Hart Decl.");

LG Electronics, Inc. v. ATI Technologies ULC, IPR2015-00325 Final Written Decision (Paper 62), PTAB, Apr. 14, 2016 (Ex. 1011, the '325 IPR)[5];

Transcript of Deposition of Dr. Massoud Pedram in ITC Investigation (Ex. 2008, "Massoud ITC Dep. Tr.");

Transcript of Hearing in Inv. No. 1318 (Ex. 2005, "ITC Hr'g Tr.");

Inv. No 37-TA-1318 Initial Determination On Violation Of Section 337 And Recommended Determination On Remedy And Bond (Ex. 2002, "ID").

[3] PCT Pub. No. WO00/63770A1, pub. Oct. 26, 2000, to Williams (Ex. 1006).
[4] U.S. Patent 5,968,167, iss. Oct. 10, 1999, to Whittaker et al. (Ex. 1007).
[5] In IPR2015-00325 ("the -00325 IPR"), the Board found claims 1, 2, and 5–7 of the '053 patent unpatentable over U.S. Patent No. 7,363,472 to Stuttard ("Stuttard '472") and the admitted prior art. The Federal Circuit reversed the Board's Decision, finding that Patent Owner had antedated Stuttard '472, which was asserted as prior art under 35 U.S.C. § 102(e). In this proceeding, Petitioner relies on the Stuttard PCT application as a prior art reference under 35 U.S.C. § 102(b). *See* Pet. 4; *see also* Ex. 1011, 13–14 (noting that the Stuttard international publication is a prior art reference under § 102(b), but that the Stuttard international publication was not cited in any of the grounds of unpatentability in the -00325 IPR).

VII. LEVEL OF ORDINARY SKILL IN THE ART

Consistent with the ID, Petitioner applies the following description of person of ordinary skill (also referred to herein as "POSITA" or "ordinarily skilled artisan"):

> a POSITA would have a bachelor degree in electrical engineering, computer engineering, computer science, or a related field, and at least two years of experience in "parallel computing or graphics processing including developing, designing, or programming software or hardware for graphics processing or parallel processing units, graphics accelerators[,] or other graphics processing systems," with additional education substituti[ing] for experience and vice versa.

Ex. 1004, Pedram Decl. ¶ 41. Patent Owner does not further address the description of a person 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). Because the description adopted in the ID and by Dr. Pedram appears commensurate with the subject matter of the '053 patent and the asserted prior art references, we apply it in the context of this Decision.

VIII. CLAIM CONSTRUCTION

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)).

At the time of the -00325 IPR, the Board gave claims their broadest reasonable construction in light of the Specification. Ex. 1011, 4. As discussed above, we now apply the *Phillips* construction standard used by U.S. district courts and the International Trade Commissions ("ITC"). Petitioner contends that the constructions adopted in the ID are actually broader than those adopted by the Board in the -00325 IPR and that applying the constructions from the ID or those from the -00325 IPR would have no meaningful effect on the outcome of this Decision. Pet. 11. Petitioner notes that, for some terms, the construction applied by the Administrative Law Judge ("ALJ") in the ID does not add limitations and no further construction is required. *See id.* at 11–12 ("arbiter only need to 'select' a command

thread"), 13 ("'relative priorities' of the 'plurality of pixel command threads and the plurality of vertex command threads' does not refer to whether the command thread is 'either a 'vertex' priority or a 'pixel' priority'), 13–14 (command engine does not require that each engine must process each command thread selected by and received from the arbiter), 14 ("memory device" given its plain and ordinary meaning), 14–15 ("coupled to" does not require direct connection). Petitioner contends that these constructions do not affect its unpatentability arguments and applies these constructions from the ID in the Petition. *See* Pet. 11.

Patent Owner does not address the above claim terms. Neither party appears to argue that any issue in this Decision turns on the above constructions. Aa we are persuaded that construction of these claim terms is not determinative, for purposes of this proceeding, , we apply the above constructions in this Decision. We now turn to constructions that appear to be disputed by the parties.

A. Command threads/select a command thread from either the plurality of pixel command threads and the plurality of vertex command threads

Petitioner states that, in the ID, the term "command threads" was construed "to mean 'a sequence of commands applicable to the corresponding element' with no requirement that the threads comprise state bits." Pet. 14; Ex. 2002, 32. According to Petitioner, "[t]his construction is sufficiently broad as to not affect Petitioner's invalidity analysis." *Id.* (citing Ex. 1004, Pedram Decl. ¶ 47).

Patent Owner does not address how the term "command thread" should be construed. Instead, Patent Owner cites a different part of the ID, where the ALJ found the term '"select a command thread' does not mean

selecting one and only one thread." Prelim. Resp. 21–22 (citing Ex. 2002, 39); *see* Ex. 2002, 41. According to Patent Owner, we should adopt the construction proposed by Petitioner and adopted in the ID, i.e., that "the claimed arbiter can satisfy the requirement of selecting 'a' command thread from the plurality of pixel and vertex command threads so long as the claimed arbiter makes a selection of *one or more* command threads of the plurality of command threads." *Id.*

Petitioner notes that, in the ID, the term "select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads" does not require an arbiter to select a command thread based on state bits associated with each command thread. Pet. 12. Patent Owner does not address this issue.

As we are persuaded that Petitioner's proposed construction of these terms is not determinative for purposes of this Decision we apply the constructions adopted in the ID for the terms "command thread" and "select a command thread."

B. At least one memory device . . . . operative to store a plurality of pixel command threads and . . . a plurality of vertex command threads

Patent Owner urges that we adopt a construction that this term requires the memory device to store complete pixel and vertex command threads, not partial command threads. Prelim. Resp. 21–22 (arguing that such a construction is consistent with Petitioner's proposed construction in the ITC Investigation and adopted in the ID; *see* Ex. 2002, 37). Petitioner does not comment on this aspect of the construction.

Patent Owner also proposes that we read into this language the requirement that the memory devices are operable to store a plurality of

pixel threads and a plurality of vertex threads at the same time. According to Patent Owner, the ID found "this is the plain language of the words in the claims, and the '053 patent specification uniformly describes the claimed memory device this way." Prelim. Resp. 22 (citing Ex. 2002, 84; Ex. 1001, 2:65–3:3, 3:40–44, 3:50–53, 3:62–4:51, 5:10–16, 5:43–56, Fig. 2, Fig. 4, Fig. 5). According to Patent Owner, in order for the arbiter to select a command thread from either the plurality of pixel command threads and the plurality of vertex command threads, as recited in claims 1 and 5, both pluralities of command threads must be present at the same time. *Id.* at 23 (citing Ex. 2001, Hart Decl. ¶ 29).

Petitioner does not address this term as a matter of claim construction, but in its unpatentability arguments notes that in the ITC Investigation Patent Owner contended that the memory device must store both pixel and vertex command threads at the same time. Pet. 25. Petitioner contends that the claim does not require the pixel and vertex threads be stored at the same time and, even if it did, the '053 patent acknowledges that simultaneous storage of pixel and vertex threads was well known in the art. As we discuss in our analysis of Petitioner's challenge, we need not construe this term because, for purposes of this Decision, we determine that Petitioner prevails under either construction.

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 1–9 As Obvious Over Stuttard And The Knowledge of a Person of Ordinary Skill In The Art As Evidenced By the Admitted Prior Art in the '053 Patent

1. *Stuttard – Exhibit 1005*

Stuttard discloses and claims a parallel processing apparatus that processes large amounts of data and observes that the computer graphics field is one example of a system that requires high speed processing of massive amounts of data. Ex. 1005, 1.[6] Stuttard discloses a "previously proposed processing architecture for processing large amounts of data in a computer system using a Single Instruction Multiple Data (SIMD) array of processing elements" and notes that "[i]n such an array all of the processing elements receive the same instruction stream, but operate on different respective data items." *Id.* at 1:24–30.

In Stuttard, the processing elements (PEs) are operably divided into a plurality of processing blocks. Ex. 1005, 2:17–23, 56:3–9 (claim 1). The processing blocks each have at least one PE and are operable to process respective groups of data items. *Id.* at 2:25–33, 56:11–18 (claims 1, 2). Within a processing block, the array of PEs operates in parallel on respective data items carrying out the same instructions (SIMD). *Id.* at 17:29–33, Fig. 8.

Figure 1 of Stuttard, shown below, is a block diagram of an exemplary graphics processing system. Ex. 1005, 3:5–6, 4:3–11.

[6] Both parties use the page numbers of the published version of Stuttard, rather than the page numbers of Exhibit 1005. For consistency, we adopt the same page numbering conventions as those used by the parties in this Decision.

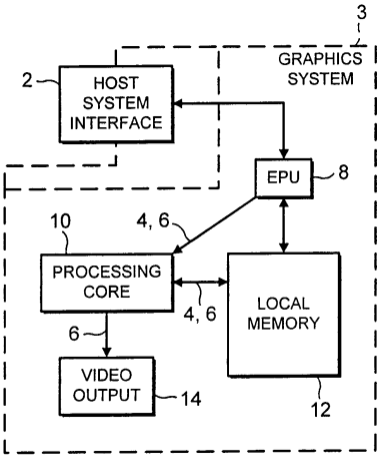
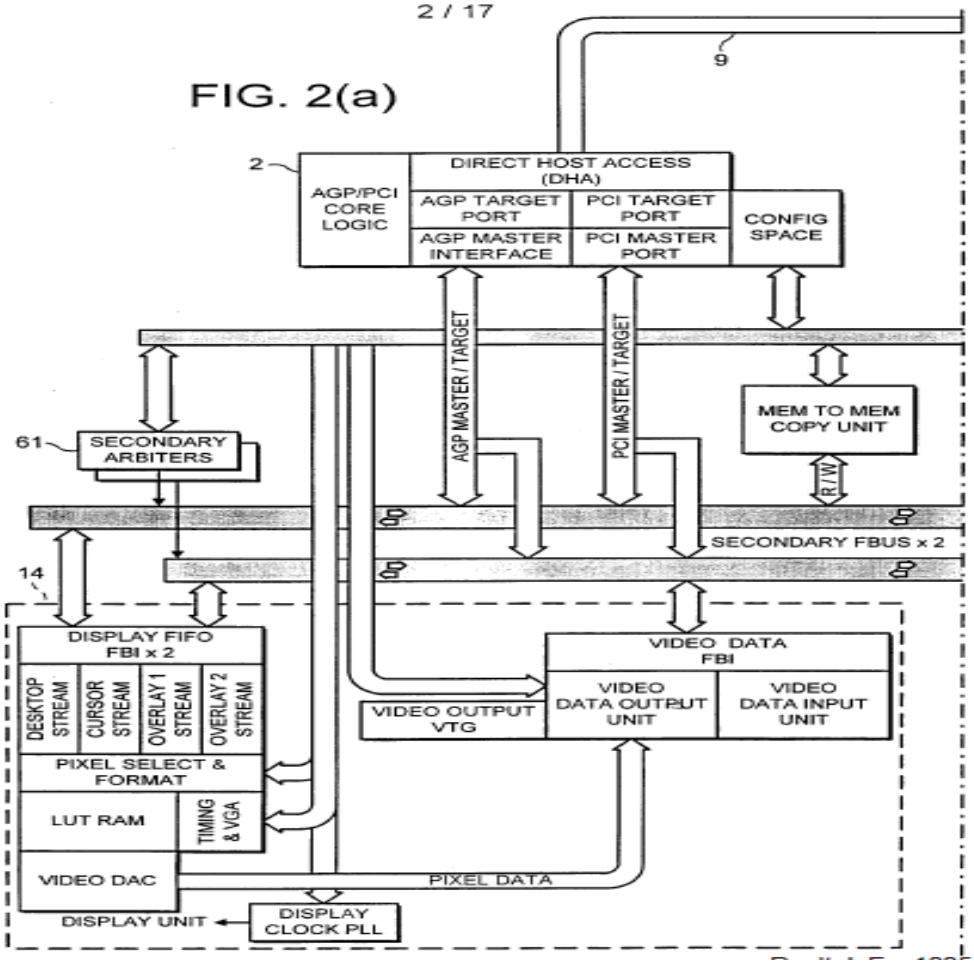


FIG. 1

Id. at Fig. 1. Graphics data processing system 3 communicates with a host system, e.g., a computer workstation (not shown), through interface 2, e.g., an advanced graphics port or PCI interface. *Id.* at 4:21–24, 5:8–28. Embedded processor unit (EPU) 8 controls overall functions of graphics processor 3. *Id.* at 4:25–30. Processor core 10 processes graphical data to output to a display screen via video output 14; local memory 12 stores information used by graphics system 3. *Id.* at 4:30–34, 7:11–16. Primary bus 4 and secondary buses 6 are also shown. *Id.* at 6:8–10.

A more detailed block diagram is shown below in Figure 2a (top), Figure 2b (bottom left), and Figure 2c (bottom right).

FIG. 2(a)


9
2
AGP/PCI CORE LOGIC
DIRECT HOST ACCESS (DHA)
AGP TARGET PORT
PCI TARGET PORT
AGP MASTER INTERFACE
PCI MASTER PORT
CONFIG SPACE
AGP MASTER / TARGET
PCI MASTER / TARGET
MEM TO MEM COPY UNIT
R / W
61
SECONDARY ARBITERS
SECONDARY FBUS x 2
14
DISPLAY FIFO FBI x 2
DESKTOP STREAM
CURSOR STREAM
OVERLAY 1 STREAM
OVERLAY 2 STREAM
PIXEL SELECT & FORMAT
LUT RAM
TIMING & VGA
VIDEO DAC
DISPLAY UNIT
DISPLAY CLOCK PLL
VIDEO OUTPUT VTG
VIDEO DATA FBI
VIDEO DATA OUTPUT UNIT
VIDEO DATA INPUT UNIT
PIXEL DATA

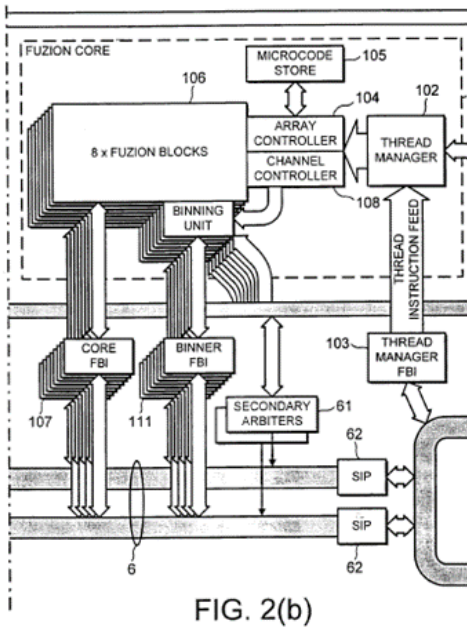


FIG. 2(b)

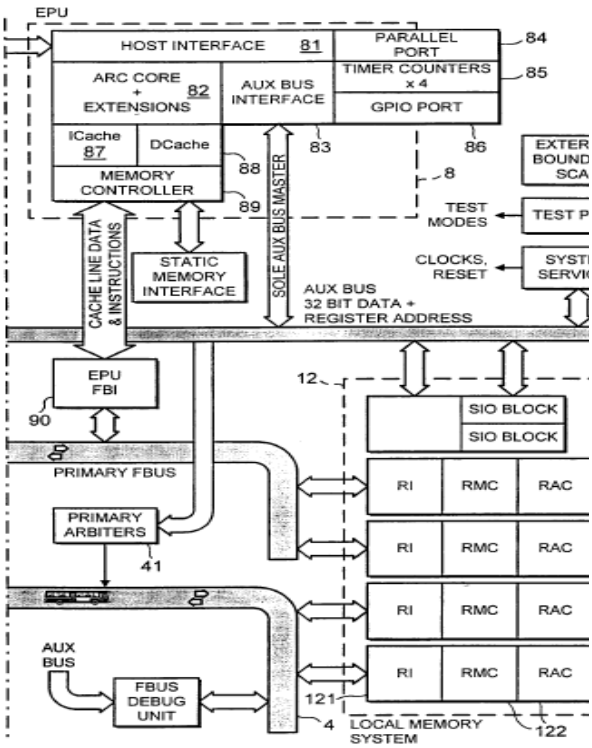


FIG. 2(c)

Ex. 1005, Figs. 2a, 2b, 2c. In Stuttard, primary bus 4 is used for connection to latency intolerant devices and secondary bus 6 is used for connection to latency tolerant devices. *Id.* at 6:6–18. The architecture accommodates any number of primary and secondary buses—the architecture in Figures 2a and 2b include 2 secondary buses. *Id.* Access to primary bus 4 is controlled by arbiter 41 and access to secondary buses 6 is controlled by secondary arbiters 61. *Id.* at 6:21–24. Secondary buses 6 are connected to primary interface bus 4 by interface units (SIP) 62. *Id.* at 6:25–27. Processing core 10, which provides the graphic processing capability of the system, is provided with 8 processing blocks 106, although any number of processing blocks can be provided in a graphics processing system of this architecture. *Id.* at 8:9–13. Core 10 is connected to secondary buses 6 by core bus interface (Core FBI) 107 for the transfer of data and binner bus interface

(Binner FBI) 111; core 10 is connected to primary bus 4 by thread manager bus interface (Thread Manager FBI) 103 for the transfer of instructions. *Id.* at 8:25–34. Local memory 12 includes memory interface units (four in this example) to communicate with local memory

Figure 3 of the Stuttard, reproduced below, shows processing core 10 in more detail. Ex. 1005, 8:17.

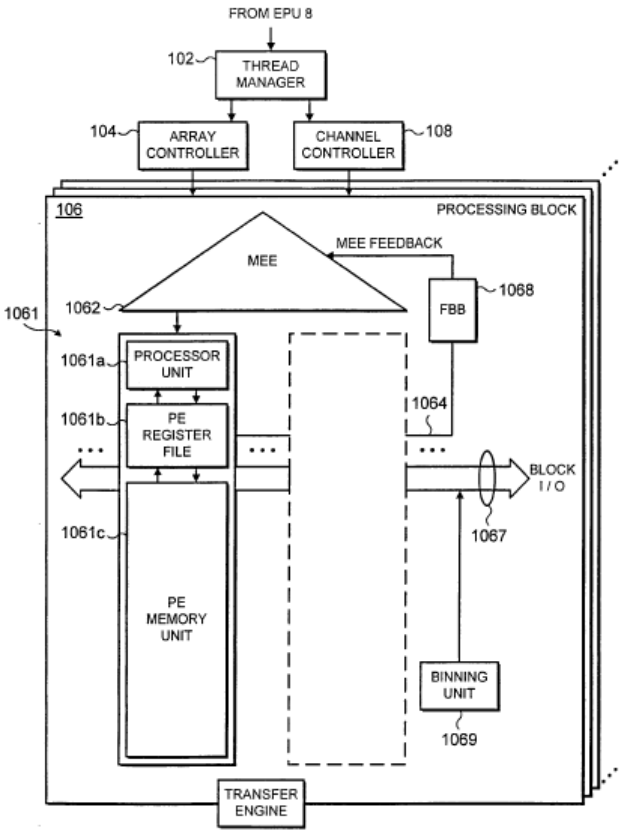


FIG. 3

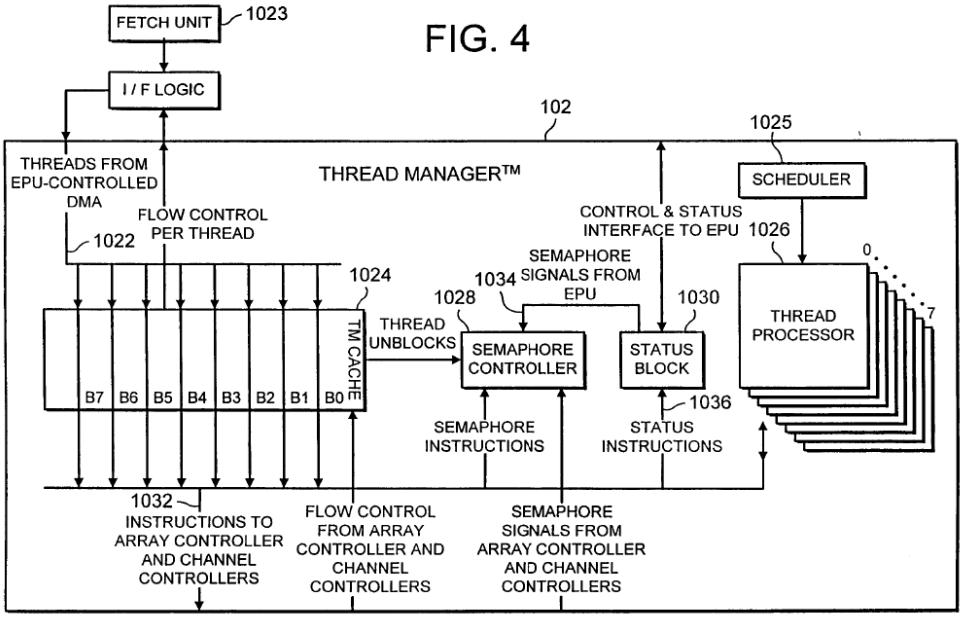
Ex. 1005, Fig. 3. Processing core 10 includes the following units controlling the operation of processing blocks 106 that perform the graphics processing: thread manager 102, array controller 104, channel controller 108, binning unit 1069 per block and microcode store 105.

Control signals from EPU 8 inform thread manager 102 about when and where instructions are to be found. Ex. 1005, 8:19–21. Thread manager

102 provides instructions to array controller 104 and channel controller 106, which transfer control signals to respective processing blocks 106 (shown one behind the other in Figure 3). *Id.* at 8:22–27. Each processing block 106 includes array 1061 of processing elements (PE(s)) and a mathematical expression evaluator (MEE) 1062, as well as binning unit 1069, FBB feedback to MEE unit 1068, and a transfer engine for controlling data transfers to and from the input/output channel under instructions from channel controller 108. *Id.* at 8:29–9:3; Fig. 8. Each PE 1061 includes processing unit 1061a for carrying out instructions received from the array controller, PE memory unit 1061c for storing data for use by PE processing unit 1061, and PE register file 1061b through which data is transferred. *Id.* at 9:11–20. Array 1061 of PEs provides a SIMD processing instruction in which array 1061 of a processing block is supplied with the same instruction, which is used to process data specific to the PE concerned. *Id.* at 9:5–9.

In Stuttard's example system, processing core 10 is provided with 8 processing blocks (*see* Ex. 1005, Fig. 2b), although any number of processing blocks can be provided in this graphics processing architecture. *Id.* at 8:9–13. Instruction streams for processing blocks 106, known as "threads," can be assigned to any task as required; threads cooperate with other threads to perform a task. *Id.* at 10:9–12, 10:23–24. Multiple threads allow a PE array to be kept alive by processing another thread when the current thread is inactive. *Id.* at 10:19–23. For example, the latency of memory I/O can be masked by assigning a plurality of threads to overlap I/O data transfer operations with processing operations. *Id.* at 10:25–24:31. Threads may also be assigned priorities relative to one another. *Id.* at 11:16–17.

Figure 4 of Stuttard, reproduced below, illustrates a thread manager in Stuttard's processing system. Ex. 1005, 3:11–12.



Thread manager 102 manages threads, which can be assigned to any task as required. *Id.* at 10:7–31, 11:19–25. Multiple threads can be used to keep a processing array active by processing another thread when the current thread is halted. *Id.* at 10:19–23. Multithreading refers to the use of several threads to perform a single task; multitasking refers to several threads performing multiple tasks simultaneously. *Id.* at 10:10–17. Thread manager 102 comprises cache memory unit 1024, which can be replaced by a series of first-in-first-out (FIFO) buffers (one per thread), instruction fetch unit 1023 (which fetches instructions for a thread from local memory or EPU 8 and supplies the instruction to cache memory 1024 via connecting logic, *id.* at 23:12–16), thread scheduler 1025 (which determines which thread is active, *Id.* at 23:18–25:23), thread processor 1026 (which provides high level control of a thread, *id* at 26:1–27:32), semaphore controller 1028 (which

synchronizes threads by controlling access to common resources, *id.* at 28:1–30:21), and status block 1030.

In the example graphics processing of Figure 9, reproduced below, data is loaded into the PEs of the graphics system so that each PE contains data for one vertex. Ex. 1005, 20:32–33.

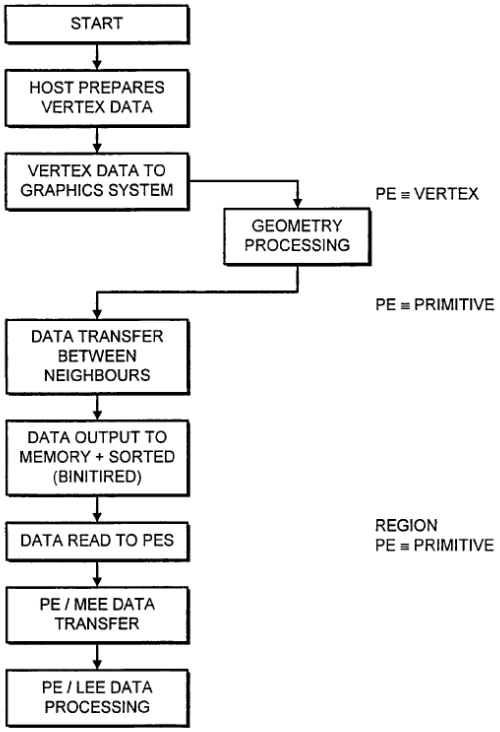


FIG. 9

The received data is then processed to transform it to the required screen space. *Id.* at 21:3–5. Each PE copies its vertex data into its neighboring PEs so that the data is organized on a primitive per PE basis (one set of vertex data corresponding to a graphical primitive). *Id.* at 21:9–13. The data is provided to binning unit 1069, which sorts the primitive data by region. *Id.* at 21:15–26. The primitive data is then written into local memory, assigned a reference appropriate to its region and read back into individual PEs. "At this stage all of the PEs in one processing block contain data concerning respective primitives occurring in a single region. From this point, a given

processing block operates on data associated with a single region of the display." *Id.* at 22:4–10. Each PE then transfers its data concerning its primitive for processing into pixel data. *Id.* at 22:12–32.

2. *Independent Claim 1*

a) Preamble 1[pre]

The preamble of claim 1 concerns a graphics processing system. Petitioner cites Figure 1 of Stuttard as illustrating a system level block diagram of a graphics data processing system 3 that interfaces with a host system (not shown) such as a personal computer or workstation via an interface 2. Pet. 28 (citing Ex. 1005, Fig. 1, 4:21–32; Ex. 1004, Pedram Decl. ¶¶ 55 – 58). Petitioner notes that graphics system 3 includes embedded processing unit (EPU) 8 that controls the overall function of the graphics processor and interfaces with the host system. *Id.* at 28–29. Petitioner also notes that graphics processing system 3 includes processing core 10, which processes graphical data for output to a display screen via video output interface 14 and local memory 12. *Id.* at 29. Patent Owner does not dispute that Stuttard discloses a graphics processing system.

b) Limitation 1.a

Petitioner identifies as claim limitation 1.a the recitation "at least one memory device comprising a first portion operative to store a plurality of pixel command threads and the second portion operative to store a plurality of vertex command threads." Pet. 29. In the context of the knowledge of a person of ordinary skill in the art, as evidenced by the admitted prior art in the '053 patent, Petitioner cites Stuttard's cache unit 1024 as teaching limitation 1.a. *Id.* at 20. Petitioner acknowledges that Stuttard does not explicitly disclose the cache unit having first and second portions operative to store pixel and vertex threads, respectively, but emphasizes that in known

systems pixel and vertex data "are often stored separately because they correspond to different types of graphics processing tasks. *Id.* at 20–21 (citing Ex. 1005, Pedram Decl. ¶ 63). Noting that Stuttard's processing block 106 processes both pixel and vertex threads and that pixel and vertex threads are different types of data, Petitioner contends that, to an ordinarily skilled artisan, storing such pixel and vertex threads separately would have been an obvious logical layout of a multithreaded processing system, because such a memory layout reduces memory access and increases operation speed. *Id.* (citing Ex. 1005, 12:1–7, 26:3–7; Ex. 1004, Pedram Decl. ¶ 63).

Petitioner cites Figure 4 of Stuttard as disclosing thread manager 102 includes cache memory unit 1024 storing instructions fetched for each thread, instruction fetch unit 1023, thread scheduler 1025, thread processors 1026, semaphore controller 1028, and status block 1030. Pet. 20–21 (citing Ex. 1005, 11: 3–14). Petitioner further cites Studdard's disclosure of replacing memory unit 1024 with a series of first-in first-out memories (FIFOs), one per thread, as evidence that an ordinarily skilled artisan would have organized pixel and vertex command threads into separate portions, e.g., as facilitated by storing the pixel and vertex threads into first and second FIFOs or two series of FIFOs. *Id.* at 21–22.

Limitation 1.a does not recite the pixel and vertex threads are stored at the same time. Under a construction that does not require the vertex and pixel information to be stored at the same time, Stuttard's disclosure of a system that performs both vertex and pixel operations separately teaches limitation 1.a.

Nevertheless, Petitioner contends that even if limitation 1.a is construed to require the pixel and vertex command threads be stored at the

same time, the prior art identified in the '053 patent demonstrates that the limitation would have been known to an ordinarily skilled artisan. Pet. 25 (citing Ex. 1001, 1:42–46, Fig. 1; Ex. 1004, Pedram Decl. ¶ 67; *see also,* Ex. 1004, Pedram Decl. ¶64).

As further evidence that Stuttard discloses storing pixel and vertex command threads at the same time, Petitioner relies on Stuttard as teaching a system with processing blocks that can operate to perform pixel and vertex processing independently. Ex. 2005, ITC H'rg Tr. 989:3–12; *see also* Ex. 2008, Pedram Dep. Tr. 149:19–152:12.

As to limitation 1.a, Patent Owner contends that Stuttard (i) does not disclose or render obvious a memory device that stores a plurality of pixel command threads and vertex command threads at the same time (Prelim. Resp. 24–33) and (ii) does not disclose or render obvious storing command threads at all (*id*. at 33–34).

We first turn to Patent Owner's second contention, i.e., that Stuttard does not disclose storing command threads at all. Prelim. Resp. 33. According to Patent Owner, Stuttard discloses only that cache 1024 stores individual instructions fetched for each thread. *Id.* With emphasis, Patent Owner quotes Exhibit 1005, page 11, lines 3–5 as reading "[t]he thread manage 102 is shown in more detail in Figure 4, and comprises *a cache memory unit 1024 for 5 storing instructions fetched for each thread"*). *Id*. This "quote" from Stuttard, taken from Dr. Hart's declaration (Ex. 2001, Hart Dec. ¶ 65) and emphasized with italics in the Preliminary Response, includes confusing text suggesting that Stuttard teaches fetching only 5 instructions for storage in local memory. The "5" in the "quoted" text is not part of Stuttard's disclosure, but instead is the designation of line number 5 on the page. Ex. 1005, 11:3–6. Patent Owner's other citations to Stuttard

merely state that instructions for a thread are fetched and supplied to a cache, but do not suggest that partial threads are fetched. Prelim. Resp. 33–34 (citing e.g., Ex. 1005, 11:12–14, 23:13–16 ("instructions for a thread are fetched from local external memory 103 or from the EPU 8 by the fetch unit 1023 and supplied to the cache memory 1024 via connecting logic").

Patent Owner's other arguments that Stuttard does not disclose fetching all the instructions for a thread are based on testimony taken out of context. In particular, Patent Owner's citation of Dr. Pedram's testimony that a cache 1024 might never store all of the instructions in a thread (Prelim. Resp. 34) fails to note that Dr. Pedram's answer, "may or may not," was given in the context of "what happens in these kinds of systems. It's called pre-fetch and execute, and you try to overlap, therefore, the actual fetch and execute operation." Ex. 2008, Pedram Dep. Tr. 183:1–10 (emphasis omitted); *see also id.* at 181:1–182:25 (providing context as to how instructions in a thread are received sequentially and begin to execute, even before subsequent instructions in the stream are fully received).

We now turn to Patent Owner's arguments that Stuttard does not disclose storing vertex command threads and pixel command threads at the same time. Patent Owner characterizes Stuttard as disclosing that all the processing blocks of the SIMD operate on a common received instruction, that the same instruction stream is supplied to all of the processing blocks in the system, and the system operates exclusively on vertices in one phase, and only when that phase is complete operates exclusively on pixels in a later phase, with no overlap in the execution of these two phases. Prelim. Resp. 13 (citing Ex. 1004, 2:25–34, 11:33–35; Ex. 2001, Hart Decl. ¶¶ 41–42). According to Patent Owner, "when Stuttard is operating in one phase, it does not store the alleged command threads relating to the other

phase in memory (in particular in the small, fast cache memory Petitioner focuses on) and would have no reason to be stored there at that time." *Id.*; *see also id.* at 25–26 (noting that, in Stuttard, pixel processing occurs after vertex processing). Patent Owner also cites to the ID's discussion of Stuttard as disclosing phases of operation, i.e., vertex operations, then binning, then pixel operations and its processing blocks, each containing an array of PEs acting "'block parallel,' which either means or strongly suggests that vertex threads run to completion before pixel threads become stored for processing." *Id.* at 26 (citing Ex. 2002, 85). Patent Owner cites the ID's conclusion, supported by the testimony of Patent Owner's expert Dr. Hart, that there would be no benefit to storing vertex and pixel threads together at the same time. *Id.* at 27 (citing Ex. 2002, 87).

Patent Owner's arguments focus on SIMD processing performed by PEs within a block, where each PE is supplied with the same instruction as the stream of instructions in the thread execute. Ex. 1005, 9:5–9. Patent Owner does not address a fundamental principle of Stuttard, i.e., the division of PEs into processing blocks.

Petitioner contends that by organizing its PEs into a plurality of processing blocks, Stuttard overcomes inefficiency associated with conventional SIMD architectures that result from SIMD arrays being limited to processing one small data item group at a time. Ex. 1004, Pedram Decl. ¶ 51 (citing Ex. 1005, 1:24–2:10). "Stuttard's system can simultaneously process multiple instructions streams with each stream executing on one processing block." *Id.* Although Stuttard provides a "particular example" in which the same instruction stream is supplied to all the processing blocks in the system Ex. 1005, 11:33–34 (referring to the example thread manager

shown in Figure 4), Stuttard explicitly discloses that different blocks may be assigned different tasks. For example, Stuttard states that

> [a]t any particular moment in time each thread may be assigned to one of any number of tasks. For example, thread 0 may be assigned for general system control, thread 1 assigned to execute 2D (two dimensional) activities, and threads 2 to 7 assigned to executing 3D activities (such as calculating vertices, primitives or rasterizing)."

Ex. 1005, 11:19–25.

Dr Pedram notes that "[w]ith multiple SIMD processing blocks, each processing block is able to process respective groups of data items" and "Stuttard's system can simultaneously process multiple instructions streams with each stream executing on one processing block." Ex. 1004, Pedram Decl. ¶ 51. Dr. Pedram also testifies that "since there are eight processing blocks in Stuttard's system (*Id.* at Fig. 2(b), and Fig. 4), each of which can execute one thread, a processing block may start the vertex processing for a newly arrived primitive while a different block is still finishing up the pixel processing for a previous primitive." Ex. 1004, Pedram Decl. ¶ 53; Ex. 2005, ITC Hr'g Tr. 995:16–996:5. Dr. Pedram's testimony is consistent with Stuttard's Figures 3 and 8, which show a binning unit as part of each processing block operating such that, after processing by the binning unit, "a given processing block operates on data associated with a single region of the display." Ex. 1005, 22:4–10.

Stuttard's discussion of a phased operation for performing vertex processing before pixel processing stands for little more than the proposition that, before one can shade or color a pixel, one must have a pixel to shade. Dr. Pedram testified Stuttard's linear graphics pipeline would perform pixel

operations after vertex operations. Ex. 2005, ITC Hearting Transcript (ITC Hr'g Tr 40:21–25). Dr. Pedram explained,

> No system would limit itself to only do vertex processing of the full image which could be a high definition image containing thousands—millions of pixels–-—do the verticis (sic) first and then start doing pixels.
>
> What they will do is to use the notion of pipelining . . .
>
> This is the idea of pipelining, functional pipelining. I do vertex processing on the top right as I'm doing pixel processing on the top left for which the vertex processing has already been completed; so at any instance in time when I just look at the data - - right? - - that's flowing in the graphics pipeline and present there, I will see both pixel and vertex data. I'll see my processing engines doing both pixel processing and vertex processing; but, of course, they are working on different data.
>
> The data is not the same. The data is different, but it's just happening at the same time so that cache memory must store the vertex data for the top right verticis (sic) and pixel data for the top left quadrant pixels.

Ex. 2008, Pedram Dep. Tr. 166:24–168:11.

Stuttard's distinction between processing elements and processing blocks is reflected in claim 1 of Stuttard which explicitly recites that the processing elements are operably divided into a plurality of processing blocks and that the processing blocks are operable to process respective groups of data items. Ex. 1005, 56:3–9, *see also id.* at 82:14–18 (claim 88). This distinction is significant in the context of the memory limitation of the independent claims of the '053 patent (limitation 1.a), which recites the memory has a first portion operative to store a plurality of pixel command threads and a second portion operative to store a plurality of vertex command threads. Ex. 1001, 7:11–14 (independent claim 1), 8:5–8 (independent claim 5).

Stuttard discloses a local memory unit segmented into portions that accommodate its processing block architecture. *See, e.g.*, Ex. 1005, 86:24–87:10, claim 103. In Stuttard's claim 103, the array of processing elements is divided into a plurality of processing blocks operable to process respective groups of data items. *Id.* at 86:26–32. Claim 103 recites the apparatus includes a local memory unit storing data items, e.g., instructions in a thread, at addresses thereof. *Id.* at 87:1–4. Claim 103 further recites a segment register for each processing block that indicates the address area of the local memory unit to be accessed by the processing block concerned. *Id.* at 87:6–10, *see also id.* at 70:27–72:15 (claims 56–59). Thus, Stuttard teaches that each processing block accesses a segment of memory appropriate for the tasks the PEs in that processing block are performing at any point in time. As Dr. Pedram notes, because Stuttard's processing blocks can carry out different tasks at the same time, e.g., vertex processing in one region and pixel processing in another region, a person of ordinary skill in graphics processing, would have understood that a first processing block carrying out pixel processing accesses that segment of memory and a processing block carrying out vertex processing accesses a different segment of memory.

Stuttard also discloses the processing elements access the local memory unit on the basis of the received address and stored memory segment information (Ex. 1005, 88:1–7, claim 104) by adding the segment information to the address information to give a local memory unit target address (Ex. 1005, 88:9–13, claim 105). As Petitioner points out, the '053 patent discloses that in a graphics engine it is well known to locate pixel command threads in one segment of memory and vertex command threads in another memory segment. Pet. 22–25 (citing Ex. 1001, 1:42–46, Fig. 1); Ex. 1004, Pedram Decl. ¶¶ 62, 63, 67). In the context of a graphics engine

implemented according to Stuttard, PEs in a processing block would access vertex command threads based on an address in one memory unit segment and pixel command threads based on an address in another memory segment.

Based on the disclosures in Stuttard, we are persuaded that Petitioner has demonstrated that Stuttard would have taught or suggested claim limitation 1.a to an ordinarily skilled artisan, whether or not the limitation is construed to require that vertex and pixel threads be stored at the same time.

c) Claim limitation 1.b

Petitioner identifies as claim limitation 1.b the recitations "an arbiter, coupled to the at least one memory device, operable to select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads based on relative priorities of the plurality of pixel command threads and the plurality of vertex command threads." Petitioner cites Stuttard's thread scheduler 1025 and thread processors 1026 in thread manager 102 as teaching the claimed arbiter. Pet. 25–26, 32–35 (citing Ex. 1005, 12:12–18, 23:35–26:32). Petitioner notes that, based on thread scheduling, thread processor 1026 assembles instructions to be issued to array controller 104 and channel controller 108 and contends that, because thread manage 102 contains cache unit 1024, Stuttard discloses that the arbiter is coupled to Stuttard's memory device. *Id.*

Patent Owner contends that Stuttard does not disclose or render obvious an arbiter that selects a command thread from a plurality of pixel command threads and vertex command threads. Prelim. Resp. 34–38. Patent Owner first contends that Stuttard's phased architecture and SIMD design executes a single instruction at a time and precludes an arbiter from

selecting between at least two pixel command threads and two vertex command threads. *Id.* at 35–36. For the reason discussed above, we are unpersuaded by Patent Owner's arguments that Stuttard's phased processing precludes storing and executing vertex threads and pixel threads at the same time.

Petitioner cites Stuttard as disclosing that cache unit 1024 can be constructed as a series of FIFO buffers, one buffer per thread. Pet. 20 (citing Ex. 1004, 11:3–10). As we are persuaded by Petitioner's arguments that in Stuttard different blocks can execute different threads, e.g., when processing different portions of an image, we also find Patent Owner's FIFO arguments are unavailing. *See* Prelim. Resp. 37–38.

d) Claim 1 Conclusion

Having considered the arguments and evidence of record, we are persuaded that, for purposed of this Decision, Petitioner has demonstrated that the limitations of claim 1 would have been taught or suggested to an ordinarily skilled artisan by Stuttard and the knowledge of a person of ordinary skill as demonstrated by the prior art in the '053 patent.

3. *Independent Claim 5*

Independent claim 5 is similar to independent claim 1, but, unlike limitation 1.b, limitation 5.b does not limit the arbiter to selecting commands based on priority. Petitioner relies on the arguments it advanced for claim limitations 1.pre, 1.a, and 1.b to support its contentions that the corresponding limitations of claim 5 are taught by Stuttard. Pet. 37–38

a) Limitation 5.c

Petitioner identifies the following recitation in claim 5 as limitation 5.c: "a plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread." Petitioner cites

Stuttard's disclosure that thread manager 102 is coupled to multiple processing blocks 106, each having multiple PE arrays 1061 and each PE array processing both vertex and command threads. Pet. 26. Petitioner contends that because claim 5 requires only that each command processing engine be operable to receive and process the thread selected by the arbiter, and the arbiter may select only pixel command threads or only vertex command threads, a command processing engine may process a pixel command thread or a vertex command thread. *Id.*

Patent Owner does not argue that Stuttard fails to disclose this limitation. For the reasons we discussed above, we agree with Petitioner and find Patent Owner's arguments unavailing.

4. *Dependent Claim 2*

Claim 2 depends from claim 1 and recites "a command processing engine, coupled to the arbiter, wherein the arbiter is further operable to provide the command thread to the command processing engine." Petitioner cites Figure 3 of Stuttard as illustrating control signals from EPU 8 directing thread manager 102 to fetch and provide instructions to array controller 104 and channel controller 108, which transfer the control signals to processing blocks 106 based on the received instructions. Pet. 35–36. Each of the processing blocks includes PEs that carry out instructions received from the array controller. *Id.*

Patent Owner does not respond explicitly to Petitioner's contentions concerning claim 2. Having reviewed the evidence of record, we are persuaded, for purposes of this Decision, that Petitioner has demonstrated Stuttard teaches the additional limitations recited in claim 2.

5. *Dependent Claims 3, 4, 8, and 9*

Dependent claims 3 and 4 depend from claim 1; dependent claims 8 and 9 depend from claim 5. For each claim Petitioner cites its discussion of the limitations of claim 1. Pet. 37, 40.

Patent Owner does not respond explicitly to Petitioner's contentions concerning claims 3, 4, 8, and 9. For purposes of this Decisions, we are persuaded that Petitioner has demonstrated the limitations of claims, 3, 4, 8, and 9 would have been taught or suggested to an ordinarily skilled artisan by Stuttard and the knowledge of a person of ordinary skill as demonstrated by the prior art in the '053 patent.

6. *Dependent Claim 6*

Claim 6 depends from claim 5 and recites "the plurality of command processing engines comprises at least one arithmetic logic unit." Petitioner cites Figure 11 of Stuttard as showing that PE 1061 includes arithmetic logic unit 214 connected to receive data values from up block of 8 bit registers 202, 204, 206, 208 via multiplexers 210 and 212. Pet. 39 (citing Ex. 1005, 47:24–30; Ex. 1004, Pedram Decl. ¶¶ 80 – 82).

Patent Owner does not respond explicitly to Petitioner's contentions concerning claim 6. Having reviewed the evidence of record, for purposes of this Decision, we are persuaded that Petitioner has demonstrated Stuttard teaches the additional limitations recited in claim 6.

7. *Claim 7*

Claim 7 depends from claim 5 and recites "the plurality of command processing engines comprises at least one texture processing engine." Petitioner cites Stuttard as disclosing a computer graphics systems in which a large amounts of data are produced that relate to, for example, geometry, texture and color of objects and shapes to be displayed on the screen.

Pet. 39 (citing Ex. 1005, 1:9–22). Petitioner further cites Stuttard as disclosing transforming data received from a host system reference space to a required screen space by processing geometry, view, lighting and shading to produce data depending upon the chosen viewpoint. *Id.* (citing Ex. 1005, 21:3–7; Ex. 1004 ¶¶ 83–84).

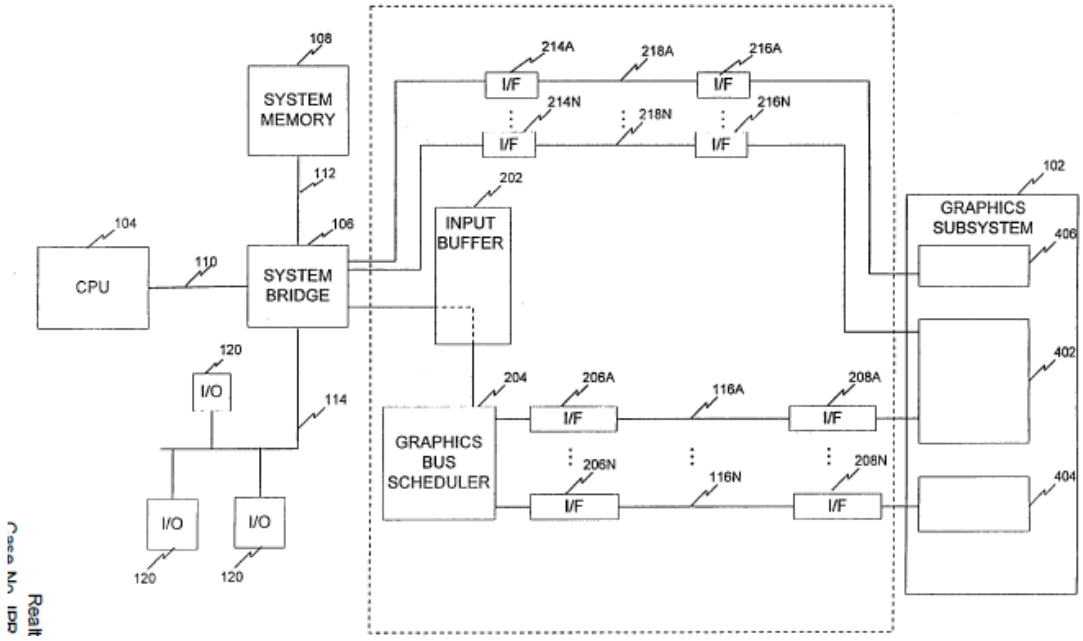
Patent Owner does not respond explicitly to Petitioner's contentions concerning claim 7. Having reviewed the evidence of record, for purposes of this Decision, we are persuaded that Petitioner has demonstrated Stuttard teaches the additional limitations recited in claim 7.

C. Petitioner's Challenge to Claims 1–9 As Obvious Over Stuttard and Williams

1. *Williams*

Williams discloses a graphics processing system that improves command transfer throughput by tagging graphics commands to indicate their specific order and assigning them to a plurality of buses according to command type (e.g., geometry, texture). Ex. 1006, (code 57), 8:4–21. In Williams, a graphics bus scheduler assigns the multiple graphics commands to the buses, so as to create a pipeline affect across the plurality of buses. *Id.* at 8:22–24. Williams states that increasing the bandwidth between a graphics subsystem and a system bridge speeds access to system memory; this reduces the required amount of local graphics memory, e.g., texture memory and/or the frame buffer, that cannot be used by other elements of the system. *Id.* at 26:1–9.

A block diagram of the embodiment in Figure 4 of Williams is reproduced below.



Ex. 1006, Figure 4

Ex. 1006, Fig. 4. Input buffer 202 temporarily stores graphics commands provided by system memory 108 in the order they are received from system bridge 106. *Id.* at 11:10–12, 11:20–22. Graphics bus scheduler ("GBS") 204 retrieves graphics commands from the input buffer and assigns them to graphics buses 116A–116N according to their type, e.g., vertex and pixel commands may be assigned to a first and second bus, 116A and 116B, respectively. *Id.* at 12:3–11, 18:26–28, 19:4–7. In the embodiment of Figure 4, based on their assigned graphics command type, GBS 204 sends the commands directly to specific functional components 402, 404 of graphics subsystem 102 that perform geometry and pixel operations, respectively. *Id.* at 22:3–9; 25:15–24. "More specifically, graphics commands can be directly transferred to the appropriate component/stage of a graphics pipeline 700." *Id.* at 22:7–8, 25:15–23.

Figure 7 of Williams reproduced below shows a graphics pipeline.

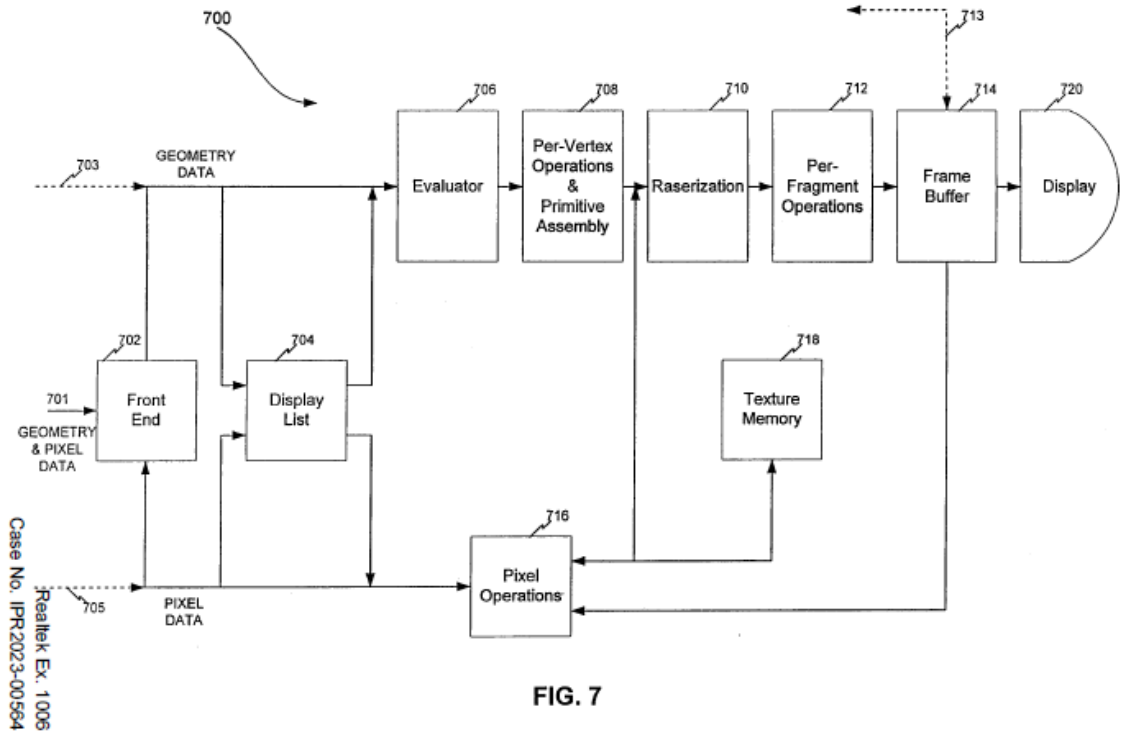


FIG. 7

Ex. 1006, Fig. 7. In Figure 7 front end 702 receives graphics commands 701 and separates them into geometry type graphics commands 703 and pixel type graphics commands 705. *Id.* at 24:11–15. In the embodiment of Figure 4, bus 116A in pipeline 700 receives geometry type graphics commands (vertices, normals, colors) 703 and bus 116 B receives pixel commands (texture) 705. *Id.* at 25:15–23. The "[g]raphics commands can be processed immediately through the pipeline 700 or can be accumulated in [the] display list 704 for processing at a later time." *Id.* at 24:16–17. Pixel related commands are provided to pixel operation stage 716 and geometry commands are provided to evaluator 706 to approximate curve and surface geometry. *Id.* at 24:12–19. Pre-vertex operations and primitive stage assembly stage 708 processes geometric primitives, such as points, line segments, triangles and polygons, all of which are described by vertices. *Id.* at 24:19–23. Vertices are then transformed and lit and primitives are clipped

to a viewport in preparation for rasterization stage 710, which produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, triangle, or polygon. *Id.* at 24:23–26. Fragments produced in the rasterization stage are fed to pre-fragment operation stage 712 to perform final operations, e.g., updating frame buffer 714 with Z values, blending incoming pixel colors with stored colors, masking, and other logical operations on pixel values. *Id.* at 24:26–25:2.

2. *Independent Claim 1*

a) Preamble, limitation 1.b

As to claim limitation 1.pre and limitation 1.b (the arbiter limitation), Petitioner relies on its arguments concerning Stuttard. Pet. 46, 48.

As to claim limitation 1.a (the memory limitation), Petitioner cites Williams as disclosing that GBS 204 retrieves graphics commands from input buffer 200 according to command type, e.g., vertex and pixel commands that are assigned to first and second buses, respectively. Pet. 41–42. Noting that the commands are stored in the order in which the GBS transfers them to their respective buses, Petitioner asserts that, to fetch and transfer the commands efficiently, to a person of ordinary skill, it would have been obvious to store different types of command in different portions of the system memory and/or the input buffer. *Id.* at 42. According to Petitioner, it was well known that commands stored in consecutive addresses may be read and written using fewer instructions than retrieving from non-consecutive addresses. *Id.*

Petitioner also cites the implementation shown in Williams' Figure 6, where vertex commands are transferred to a graphics de-scheduler and an

output buffer before being routed to the graphics subsystem, while pixel commands are transferred to and stored in a separate input buffer. Pet. 43.

Petitioner contends that an ordinarily skilled artisan would have had reason to combine the teachings of Stuttard and Williams because they both involve processing of conventional graphics objects and the combination would apply existing technology with predictable results to achieve a shared objective, i.e., to improve processing speed in a high throughput command architecture. Pet. 44–45. Petitioner further asserts that an ordinarily skilled artisan would have understood that vertex and pixel commands are used in distinct stages in graphics processing pipelines, such that storing vertex and pixel commands in separate storage would have been an obvious design choice. *Id.* at 44 (citing Ex. 1005, 20:21–22:32; Ex. 1006, Fig. 7; Ex. 1004, Pedram Decl. ¶ 99).

Noting that Williams does not explain the relationship between multiple commands, Patent Owner contends that Williams describes storing only individual commands, but does not describe storing sequences of commands, i.e., threads, in the input buffer. Prelim. Resp. 39–41. According to Patent Owner, because Williams uses multiple buses and the GPU processes commands in the order they are received from the CPU, Williams demonstrates single threaded execution and a person of ordinary skill would not have understood such execution to demonstrate multi-threaded execution. *Id.* at 40–41.

Petitioner cites the pipelining shown in Figure 7 of Williams. Pet. 44. Williams explains that vertex and pixel commands are received on buses 116A and 116B "[g]raphics commands can be processed immediately through the pipeline 700 or can be accumulated in the display list 704 for processing at a later time." Ex. 1006, 24:16–17. The accumulation of such

commands demonstrates that Williams stores sequences of vertex and pixel commands, i.e., threads, at the same time. Moreover, as shown in Figure 7 Williams teaches processing these threads in different processing paths. Although storing vertex commands and pixel commands in separate contiguous portions of memory would appear to be a logical design choice in such a system, limitation 1.a does not limit the size or location of the first and second portions of memory and does not require the commands be stored contiguously.

Patent Owner also argues that the Petition effectively concedes Williams does not disclose limitation 1.a, instead asserting it would have been obvious to improve processing efficiency by storing vertex and pixel commands in dedicated portions of memory with consecutive addresses. Prelim. Resp. 41–42. This argument by Patent Owner merely repeats the arguments advanced by Petitioner as to why such storage would have been an obvious expedient. According to Patent Owner, modifying Williams to read particular memory addresses would substantially slow the system down and render the GBS units, which sort commands by type, redundant. *Id.* at 43. We note that the GBS sorts the commands before transmitting them to a bus interface, i.e., the GBS merely recognizes which commands are geometry commands destined for bus 116A and which commands are pixel commands destined for bus 116B. Williams' GBS does not implicate the efficiency of storing vertex and pixel commands in separate portions of the input buffer. We further note that limitation 1.a does not limit the first and second memory portions to portions with contiguous addresses and would not exclude a system where the GBS sorts the commands.

Responding to Petitioner's arguments concerning Williams' Figure 6 implementation, Patent Owner contends that Williams does not state the

pixel and vertex commands are stored at the same time and that Williams' input and output buffers are not coupled to an arbiter that selects a command thread from either the plurality of pixel command threads or vertex command threads. Prelim. Resp. 43–44. In this ground, Petitioner cites Stuttard as disclosing the claimed arbiter coupled to the memory device. Pet. 48.

Patent Owner contends a person of ordinary skill would not replace Stuttard's cache unit with Williams' system memory, input buffer and output buffer because their function is redundant of Stuttard's EPU/local memory, which also acts as a bridge between the host and graphics systems and has storage for command threads. Prelim. Resp. 45–46. Petitioner does not propose combining Stuttard's and Williams' systems in a manner that would be redundant. Petitioner contends that a person of ordinary skill would have understood that Williams' input buffer could be used in Stuttard and that it would have been an obvious expedient to organize the buffer into separate portions for vertex and pixel threads that the GBS would recognize and route to the appropriate bus. Petitioner also points to the pipeline in Figure 7, where pixel and vertex threads can be accumulated, i.e., stored, in the display list.

Patent Owner also argues a person of ordinary skill would not have combined the teachings of Stuttard and Williams because the GBS sorts commands by type. Prelim. Resp. 46–47. Patent Owner acknowledges that batching commands in memory may be useful if a system is capable of transferring a batch of data, but that Williams does not teach that function. *Id.* at 47–48. We note, however, that Petitioner points out Williams teaches a pipeline that performs vertex and pixel processing along separate paths in

which vertex and pixel commands are used in different stages of graphics processing. Pet. 44.

In consideration of the above, for purposes of this Decision, we are persuaded that Petitioner has cited sufficient evidence to demonstrate that a person of ordinary skill would have had reason to combine the teachings of Stuttard and Williams and that in combination the references teach all the limitations of claim 1.

3. *Independent Claim 5*

Petitioner cites Stuttard as disclosing limitations 5.pre, 5.b. and 5.c for the same reasons as those discussed above. Pet. 49–50; *see* Section IX.B herein. Petitioner cites Williams as disclosing limitation 5.a for same reasons as those presented with respect to limitation 1.a. *Id.* at 49; *see* Section IX.C.2 herein.

Patent Owner does not address claim 5 specifically. For the same reasons as those discussed above, we are persuaded that Petitioner has demonstrated a person of ordinary skill would have had reason to combine the teachings of Stuttard and Williams and that in combination the references teach all the limitations of claim 5.

4. *Dependent Claims 2, 6, and 7*

Petitioner cites its previous discussion of Stuttard and the knowledge of one of ordinary skill in the art as evidenced by the prior art discussed in the '053 patent as the basis of its challenge to claims 2, 4, and 6 under this ground. Pet. 48–50. For the same reasons as those discussed above, we are persuaded that Petitioner has demonstrated the combined teachings of Stuttard and Williams would have disclosed or suggested the limitations of claims 2, 6, and 7 to an ordinarily skilled artisan.

5. *Claims 3 and 8*

Claims 3 and 8 depend from claims 1 and 5, respectively, and recite that "the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device." Petitioner cites Williams as disclosing GBS 204 transfers pixel (texture) related graphics commands across bus 116A to an appropriate functional component of graphics subsystem 102 and geometry related graphics commands across buses 116B – 116D to GBD 210. Pet. 49 (citing Ex. 1006, 23:11–14; Ex. 1004, Pedram Decl. ¶¶ 109–111). Dr. Pedram testifies that, in Williams, only geometry or vertex related graphics commands are sent to output buffer 212 for reordering, where pixel commands are sent directly from input buffer 202 to graphics subsystem 102. Ex. 1004, Pedram Decl. ¶ 110. Dr. Pedram states that "this means that the input buffer 202 and the output buffer 212 would comprise two separate memory devices for each portion claimed in element 1[a]." *Id.*

Patent Owner does not address claims 3 and 8 specifically. Based on the evidence and arguments of record, for purposes of this Decision, we are persuaded that Petitioner has demonstrated a person of ordinary skill would have had reason to combine the teachings of Stuttard and Williams and that in combination the references teach all the limitations of claims 3 and 8.

6. *Claims 4 and 9*

Claims 4 and 9 depend from claims 1 and 5, respectively, and recite that a single memory device comprises the first portion and the second portion of the at least one memory device. Petitioner cites its discussion of claim limitation 1.a under this ground as evidence that the combination of Stuttard and Williams discloses this limitation. Pet. 49–50. For the reasons

discussed above with respect to claim limitation 1.a, we agree with Petitioner as to claims 4 and 9.

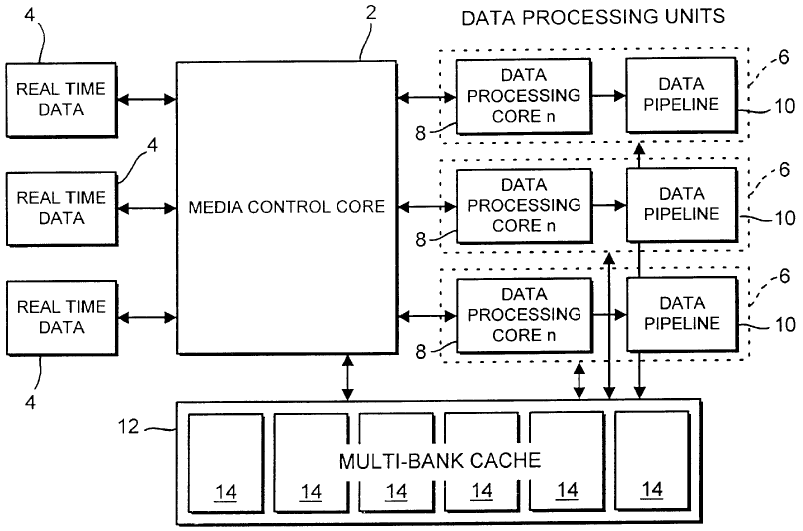
D. Petitioner's Challenged to Claims 1–9 As Obvious Over Williams and Whittaker

1. *Whittaker (Ex. 1007)*

Whittaker is drawn to a multi-threaded data processing management system. Ex. 1007 (code 54). Whittaker's processing management system, which includes a media control core (MCC), a number of data processing units, and a multibank cache, controls the execution of multiple threads of processing instructions, such as instructions that are employed to process multimedia data. *Id.*, (code 57). When processing instructions for each thread, multimedia core identifies the data processing operation to be executed as well as the resources needed to execute the operation. *Id.* For each instruction in a thread, the MCC determines if resources are available to execute the operation. *Id.* If the resources are available, the MCC identifies the operation with the highest priority and passes that to the data processing units for execution. *Id.*

Figure 1 of Whittaker, reproduced below, is a block diagram of the architecture of an embodiment. Ex. 1007, 3:36–37.

FIG. 1



Id. at Fig. 1. MCC 2 is a multimedia processing unit that directs data from and to input/output devices 4, e.g., video sources, video output, audio sources, audio outputs. *Id.* at 3:36–44. Processing units 6 consisting, for example, of data pipelines 10, are under the control and scheduling of MCC 2. Multibanked cache 12 is divided into a plurality of programmable banks 14 that can be programmed to match the requirement of one of the data processing tasks, e.g., caching texture maps for use in 3D graphics rendering. *Id.* at 4: 44–55. For example, one bank may be used for command data, another for 3D texture maps, and a third for 2D parameters. *Id.* at 9:62–66. Using multiple cache banks allows the cache to be non-blocking, i.e., if one cache is dealing with a request that it cannot satisfy, e.g., data for a read instruction is not available, another processing thread using a separate cache may be run. *Id.* at 4:56–60.

Figure 6, reproduced below, illustrates how resource checking and process selection is performed by the MCC. Ex. 1007 at 3:29–31, 8:17–40.

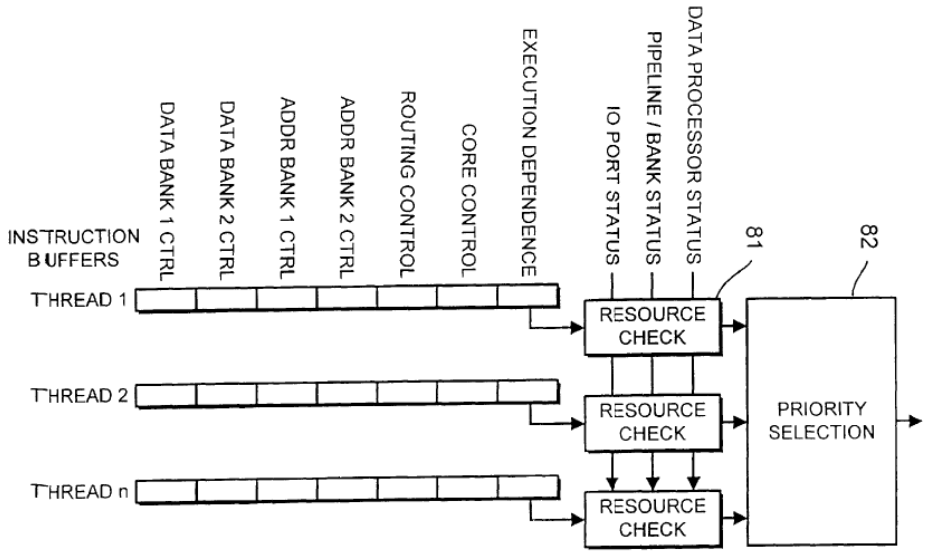


Figure 6 of Whittaker

Figure 6 illustrates three threads. *Id.* at 8:17–18. The next instruction for a thread is provided in an instruction buffer, such that the instruction is always available for resource checking and priority selection and there is no loss of execution time by checking the status every clock cycle. *Id.* at 8:36–40. For each of the possible threads, global status information received from the databanks, routing control data from the control unit, control status information from the control unit 16, and execution dependency data from other processes are sent to resource checker 81. *Id.* at 8:18–24. Resource checker 81 combines that data with various pipeline databanks status information and the status of various data processing units to determine if it is possible to run that data processing thread. *Id.* at 8:24–27. If it is possible to run the data processing thread, information about the priority of that data processing thread is used to select the processing thread to be executed; for example, real time data input, such as a video, would be given a high priority and would take precedence over a background processing operation. *Id.* at 8:27–34.

2. *Independent Claim 1*

Petitioner cites Williams as disclosing claim limitations 1.pre, and 1.a (the memory limitation) for the same reasons as those discussed above. Pet. 56; *see* Section IX.C. herein.

As to the arbiter limitation (limitation 1.b), Petitioner acknowledges that Williams does not explicitly disclose that its GBS schedules or selects threads based on relative priorities. Pet. 51. Petitioner argues that Whittaker demonstrates that this type of priority selection was well known in the graphics processing systems art. *Id.* (citing Ex. 1004, Pedram Decl. ¶ 122). Petitioner cites Figure 1 of Whittaker as illustrating a graphics processing system in which command threads are routed to MCC 2 that checks which, if any, of the possible operations have the resources to be executed and, of those, which has the highest priority. *Id.* at 51–52 (citing Ex. 1007, 3:60–63; Ex. 1004, Pedram Decl. ¶¶ 123, 125). Citing Figure 6 of Whittaker, Petitioner states in MCC 2, instruction buffers feed stored threads 1–3 and their status information to resource checkers 81, which route threads with all available resources to priority selector 82. *Id.* at 52–53 (citing Ex. 1007, 3:60–63, 3:66–67, 8:17–23, 8:28–32; Ex. 1004, Pedram Decl. ¶¶ 123, 135, 136–137). Petitioner asserts that, a person of ordinary skill would have recognized that the priority information of Whittaker could be provided to Williams' GBS, which assigns command threads to buses, to facilitate making such assignments based on priority. *Id.* at 52–53.

Petitioner contends that an ordinarily skilled artisan would have had reason to combine the teachings of Williams and Whittaker because both concern conventional 3D graphics using vertex and pixel threads, as well as parallelism related to multi-thread processing thread selection functions. Pet. 54 (citing Ex. 1006, 18:26–28, 19:4–7, Ex. 1007, 3:28–30. Petitioner

further notes that Williams and Whittaker both offer solutions relating to resource selection and improving 3D graphics processing and a person of ordinary skill would have had a reasonable expectation of success in combining their teachings. *Id.*

Patent Owner first argues that the Petition is procedurally incomplete because in asserting the combined teaching in Stuttard and Williams (Ground 2) Petitioner did not provide an element-by-element analysis for Williams, relying on Williams to supplement only the memory device limitation (limitation 1.a). Prelim. Resp. 48–49. According to Patent Owner, in this ground asserting Williams and Whittaker (Ground 3), Petitioner does not explain how each independent claim limitation is met by Williams and/or Whittaker and instead improperly relies on claim charts. *Id.* at 49 (citing Pet. 56–61). As discussed below, the Petition sufficiently explains Petitioner's arguments.

Referencing its discussion of Ground 2, Patent Owner contends that Williams does not disclose or render obvious storing command threads (Prelim. Resp. 50–51). We addressed that issue in our discussion of Petitioner's challenge based on Stuttard and Williams (Ground 2) and found Patent Owner's arguments unpersuasive. *See* Section IX.C.2 herein.

Patent Owner also contends that the Petition fails to demonstrate that Williams discloses or renders obvious an arbiter that selects a command thread. Prelim. Resp. 51–52. Observing that, in its claim chart, Petitioner appears to assert that Williams' GBS 202 is the claimed arbiter, Patent Owner argues that the GBS does not select graphics commands. *Id.* at 51. Again, referencing its response to Ground 2, Patent Owner argues that the GBS receives commands one by one from the input buffer and sorts them, but does not select commands. *Id.* (citing Ex, 1006, 18:26–28; Ex. 2001,

Hart Decl. ¶ 89.) As discussed above in the context of a graphics pipeline, we are not persuaded by Patent Owner's argument. *See* Section IX.C.2 herein.

Patent Owner further asserts that neither Williams nor Whittaker discloses or renders obvious an arbiter that selects command threads based on relative priorities of command threads. Prelim. Resp. 54–56. According to Patent Owner, Whittaker does not disclose pixel or command threads at all, let alone storing them in separate portions of memory and selecting them. *Id.* at 54 (citing Ex. 2001, Hart Decl. ¶ 94). On this point, the Preliminary Response and Dr. Hart's declaration read almost the same, but there is insufficient explanation to support this assertion. Whittaker extensively discusses processing threads according to available resources and priorities. *See, e.g.,* Ex. 1007, 4:44–61 (describing multibanked cache 12 having plurality of cache banks 14, e.g., one cache bank being dedicated to caching texture maps, and disclosing that "if one of the cache banks is dealing with a request which it is currently unable to satisfy . . . then another processing thread which uses a separate cache bank may be run"); *see also id.* at Fig. 6. Thus, Patent Owner's argument is unavailing.

Patent Owner contends that there is no motivation to combine Williams with Whittaker, nor is there any reasonable expectation of success, arguing that Williams targets high performance graphics through bus optimization and Whittaker seeks to achieve a less costly multimedia processor by prioritizing threads to be processed. Prelim Resp. 56–57. Not only does Patent Owner's argument fail to address a lack of motivation to combine, Patent Owner's argument suggests still another reason an ordinarily skilled artisan would have combined the teachings of Williams and Whittaker.

In support of its contention that an ordinarily skilled artisan would not have reasonably expected the combination of Williams and Whittaker to be successful, Patent Owner argues Williams' scheduling of commands to be transmitted over different buses conflicts with Whittaker's design, which ignores media types and uses a single MCC that bottlenecks the connection. Prelim. Resp. 57–58. We are not persuaded by Patent Owner's argument, as the purpose of checking resource availability and allocating resources based on availability and priority in Whittaker is equally applicable to allocating bus access in Williams.

For the reason discussed above, we are persuaded, for purposes of this Decision, that Petitioner has demonstrated a person of ordinary skill would have had reason to combine the teachings of Williams and Whittaker and that their combined teachings would have disclosed or suggested all the limitations of claim 1.

3. *Claims 2 and 5*

Claim 2 depends from claim 1 and further recites "a command processing engine, coupled to the arbiter, wherein the arbiter is further operable to provide the command thread to the command processing engine." Petitioner cites Williams as disclosing that GBS 202 sorts graphics commands by type, and assigns and transfers the commands on buses 116 to graphics subsystem 102; at subsystem 102 the commands are provided to functional components 402, 404, e.g., components that perform graphics and pixel processing. Pet. 58–59 (citing Ex. 1006, 22:3–7; Ex. 1004, Pedram Decl. ¶ 140).

Petitioner cites its arguments concerning claim 1 as teaching the preamble and limitations 5.a–5.b. As to limitation 5.c, which recites "a plurality of command processing engines, coupled to the arbiter, each

operable to receive and process the command thread," Petitioner cites its arguments in this ground concerning claim 2.

Patent Owner does not respond explicitly to Petitioner's contentions concerning claim 2. Patent Owner notes that claim 5 differs from claim 2 in that it requires more than one processing engine operable to receive and process the command thread. Prelim Resp. 52. Referencing Petitioner's arguments concerning claim 2, Patent Owner argues that Petitioner identifies functional components 402, 404 as the claimed processing engines, but fails to demonstrate they are coupled to alleged arbiter 204. *Id.* at 53. Based on Williams' disclosure that functional components 402, 404 perform geometry and pixel operations respectively, Patent Owner argues that the functional components are not processing engines because they are type specific components. *Id.* at 53–54 (citing Ex. 2001, Hart Decl. ¶ 93, Ex. 2005, ITC Hr'g Test. 988:8–19. The Preliminary Response and Dr. Hart's Declaration read almost the same—it is Dr. Hart who cites the testimony of Dr. Pedram at the ITC Hearing. Dr. Pedram's testimony, cited by Dr. Hart and in the Preliminary Response, states only that a processing engine must be operative to process whichever thread is provided to it by the claimed arbiter. Ex. 2005, ITC Hr'g Test. 988:8–19.

Williams does not state that functional blocks 402, are type specific, only that they can perform the type of processing required, i.e., they "can be functional components of graphics subsystem 102 that perform geometry and pixel operations, respectively." Ex. 1006, 22:5–7. Nothing about this language requires the functional components to be type specific—it simply states that the functional components are capable of processing the information sent to them.

For the reason discussed above, we are persuaded that Petitioner has demonstrated a person of ordinary skill would have had reason to combine the teachings of Williams and Whittaker and that their combined teachings would have disclosed or suggested all the limitations of claims 2 and 5.

4. *Claims 3 and 8*

Petitioner cites its discussion of claim 3 and 8 in its challenge based on Stuttard and Williams as the basis for its reliance on Williams in this ground. Pet. 59, 61. For the reasons discussed above we are persuaded, for purposes of this Decision, that Petitioner has demonstrated that in combination Williams and Whittaker would have taught the limitations of claims 3 and 8 to an ordinarily skilled artisan.

5. *Claims 4 and 9*

Petitioner cites its discussion of claim limitation 1.a as demonstrating the limitations of claims 4 and 9 are taught by the combination of Williams and Whittaker. Pet. 59, 61. For the reasons discussed above, we are persuaded, for purposes of this Decision, that in combination Petitioner has demonstrated Williams and Whittaker would have taught the limitations of claims 4 and 9 to an ordinarily skilled artisan.

6. *Claim 6*

Claim 6 depends from claim 5 and recites the plurality of command processing engines comprises at least one arithmetic logic unit. Petitioner cites Williams as disclosing geometry calculations can be performed by CPU 104 or, in the alternative, a local processor of graphics subsystem 102. Pet. 60 (citing Ex. 1006, 2:26–30). Petitioner further cites evaluator 706, which approximates curve and surface geometry by evaluating polynomial commands of input values, primitive assembly stage 708, which processes geometric primitives, and fragment operation stage 712, which performs

final operations on graphics data before the data is stored as pixels in frame buffer 714. *Id.* (citing Ex. 1006, 24:18–23; 24:27–25:2).

Having reviewed the evidence and arguments of record, for purposes of this decision, we are persuaded that Petitioner has demonstrated Williams and Whittaker in combination would have taught the limitations of claim 6 to an ordinarily skilled artisan.

7. *Claim 7*

Claim 7 depends from claim 5 and recites "the plurality of command processing engines comprises at least one texture processing engine." Petitioner cites Williams as disclosing "graphics subsystem 102 processes texture data in order to create lifelike surfaces and shadows within a 3D image." Pet. 61 (citing Ex. 1006, 3:1–3; Ex. 1004, Pedram Decl. ¶ 149).

Having reviewed the evidence and arguments of record, for purposes of this decision, we are persuaded that Petitioner has demonstrated Williams and Whittaker in combination would have taught the limitations of claim 7 to an ordinarily skilled artisan.

X. CONCLUSION

For the reasons discussed above, we determine that Petitioner has demonstrated a reasonable likelihood that it will succeed on each of its challenges to patentability, in particular:

> Claims 1–9 as Obvious Over Stuttard And The Knowledge of a Person of Ordinary Skill In The Art As Evidenced By the Admitted Prior Art in the '053 Patent;
>
> Claims 1–9 As Obvious Over Stuttard and Williams; and
>
> Claims 1–9 As Obvious Over Williams and Whittaker.

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 '053 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.

FOR PETITIONER:

Jeffrey Johnson
Christopher J. Higgins
ORRICK, HERRINGTON & SUTCLIFFE, LLP
3j6ptabdocket@orrick.com
0chptabdocket@orrick.com
Realtek-AMD_OHS@orrick.com

Steve Baik
WHITE HAT LEGAL
sbaik@whitehat.legal

FOR PATENT OWNER:

William A. Meunier
Michael T. Renaud
Adam S. Rizk
MINTZ, LEVIN, COHN, FERRIS, GLOVSKY & POPEO, P.C.
wameunier@mintz.com
mtrenaud@mintz.com
arizk@mintz.com