

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,
v.

ATI TECHNOLOGIES ULC
Patent Owner.

Case No. IPR2023-00564

U.S. Patent No. 7,742,053

**PETITIONER'S REPLY TO PATENT OWNER'S
RESPONSE TO PETITION**

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. CLAIM CONSTRUCTION	1
A. “command threads”	1
B. “at least one memory device ... operative to store a plurality of pixel command threads and ... operative to store a plurality of vertex command threads”	2
C. “select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads”	3
D. “command processing engine”	3
III. GROUND 1: STUTTARD IN VIEW OF THE KNOWLEDGE OF A POSITA	4
A. “At The Same Time” Is Not A Claim Limitation	4
1. Stuttard Is Not Limited to Phased Operation	6
2. Stuttard Operates on More Than One Instruction at a Time	7
3. Stuttard Need Not Care About Command Type to Run Multiple Types	10
4. “Operative to” Requires Only “Capability”	10
B. Stuttard Discloses And Renders Obvious Storing “Complete” Command Threads Even Though Not Required by the Claims	11
C. Stuttard Renders Obvious the Arbitrator Claim Limitation	12
D. Stuttard Renders Obvious A Texture Processing Engine	14
IV. GROUND 2: STUTTARD IN VIEW OF WILLIAMS AND THE KNOWLEDGE OF A POSITA	14
A. Storage of command threads in first and second portions of memory	15
B. Stuttard Stores “command threads”	17
C. Stuttard Satisfies the Claimed Arbitrator	18
D. A Second Memory is an Obvious Design Choice Within the Knowledge of Skill in the Art, Which Williams Supports	18
E. Motivation to Combine and Reasonable Expectation of Success	18

V.	GROUND 3: WILLIAMS IN VIEW OF WHITTAKER AND THE KNOWLEDGE OF A POSITA.....	20
A.	Storage of Pixel or Vertex Command Threads	20
B.	Williams Discloses or Renders Obvious the Claimed Arbiter.....	20
C.	Claims 2, 5-9 – Williams’ Command Processing Engines Process the Command Thread They Receive from the Claimed Arbiter.....	22
D.	Claim 6 – Williams’ Disclosure Renders Claim 6 Obvious	23
E.	Claim 7 – Williams Renders Obvious a Texture Processing Engine..	23
F.	Motivation to combine and reasonable expectation of success	23

PETITIONER'S EXHIBIT LIST

Exhibit #	Reference Name
1001	U.S. Patent 7,742,053 to Lefebvre et al.
1002	Prosecution History of U.S. Patent 7,742,053
1003	[Intentionally Omitted]
1004	Declaration of Massoud Pedram, Ph.D.
1005	PCT Pub. No. WO00/62182A2 to Stuttard et al.
1006	PCT Pub. No. WO00/63770A1 to Williams et al.
1007	U.S. Patent 5,968,167 to Whittaker et al.
1008	Curriculum Vitae of Massoud Pedram, Ph.D.
1009	[Intentionally Omitted]
1010	Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same, Inv. No. 337-TA-1318 (Order No. 31, Oct. 18, 2022)
1011	LG Elecs., Inc. v. ATI Techs. ULC, IPR2015-00325, Paper 62 (Final Written Decision) (PTAB Apr. 14, 2016)
1012	U.S. Patent 7,116,332 to Boyd et al.
1013	U.S. Patent 7,038,685 to Lindholm et al.
1014	H. Selzer, Dynamic Load Balancing Within a High Performance Graphics System, Proceedings of the Sixth Eurographics Conference on Advances in Computer Graphics Hardware: Rendering, Visualization and Rasterization Hardware (Sept. 1, 1991)
1015	Declaration of Dr. Pedram
1016	Deposition Transcript of Dr. John Hart

TABLE OF AUTHORITIES

Page(s)

CASES

<i>In re Imes</i> , 778 F.3d 1250 (Fed. Cir. 2015)	10
<i>Intel Corp. v. Qualcomm Inc.</i> , 21 F.4th 801 (Fed. Cir. 2021)	3
<i>ParkerVision, Inc., v. Qualcomm Inc.</i> , 903 F.3d 1354 (Fed. Cir. 2018)	10

OTHER AUTHORITIES

37 C.F.R. § 42.6(a).....	27
37 C.F.R. § 42.6(e).....	26
37 C.F.R. § 42.24(a).....	27
<i>ATI Techs. ULC v. Iancu</i>	4

I. INTRODUCTION

The Petition shows that claims 1-9 (the “Challenged Claims”) of U.S. Patent No. 7,742,053 (“the ’053 patent”) are unpatentable. ATI’s Response (“POR”) does not overcome the Petition’s evidence and arguments as set forth below.

II. CLAIM CONSTRUCTION

A. “command threads”

No construction is necessary. The prior art¹ does not disclose that only “partial” instructions or threads are stored. For example, Stuttard states “instructions for a thread are fetched ... and supplied to the cache memory.” Ex. 1005, 11:12-14, 23:13-16. Regardless, a POSITA would have found it obvious that the memories in the prior art (and Admitted Prior Art) are capable of storing complete command threads at any given moment in time, which is all that is required. *See, e.g.*, Ex. 2008, 183:6-10 (Pedram testified that Stuttard’s cache unit may store all of the instructions in a thread.); *supra* at Section III.A.4. (“operative” means “capable”).

Moreover, Dr. Hart lacks any credibility on this issue. Dr. Hart testified that a command thread is “a sequence of commands applicable to the corresponding element.” Ex. 1016 31:17-19. But when asked if storing the entire sequence of commands meant storing a complete command thread, Dr. Hart refused to agree. *Id.* 31:20-34:21. When asked what the claim required to be stored, Dr. Hart testified that

¹ “Prior art” means the prior art cited in this action, including the Admitted Prior Art and knowledge of a POSITA.

he never considered it. *Id.* 34:22-35:7. He then shifted and testified that only “some representation of the complete command thread” must be stored, a completely new theory. *Id.* 35:16-22.

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

ATI’s proposed construction improperly requires both pluralities of threads to be stored in memory at the same time. ATI’s arguments rest on (1) importing limitations from the specification, (2) its flawed interpretation of the arbiter claim language (that it must select from at least two pixel and two vertex command threads at the same time), and (3) its incorrect interpretation that “operative to” means *more than* “capable of.”

The plain language of the claim does not require that both pluralities need be present “at the same time.” ATI does not and cannot show any clear and unmistakable disavowal from this plain meaning. The arbiter claim limitation also does not require both pluralities to be present at the same time. *See* Section II.C.. The term “operative to” means only “capable of,” not “configured to.” *See* Section III.A.4.

C. “select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads”

ATI’s construction asks the Board to remove “either of” from the limitation. Constructions that render terms meaningless or superfluous are highly disfavored. *Intel Corp. v. Qualcomm Inc.*, 21 F.4th 801, 810 (Fed. Cir. 2021).

Contrary to ATI’s suggestion, Petitioner’s interpretation does not render any part of the claim meaningless. The plain language also covers selecting from either plurality of command threads, which means that only one plurality must be present at the time of selection. Ex. 2007, 214:20-24, 215:6-19. Indeed, dependent claims 3 and 8 show that the memory device of claims 1 and 5 (respectively) may be two separate devices, putting the pixel and vertex command threads in two different locations. The arbiter may decide to only select from the first device (storing pixel threads) without ever considering the second device (storing vertex threads) and still be within the claims. Later the arbiter may select from the second device. The arbiter is still capable of selecting from either plurality.

D. “command processing engine”

Nothing in the claims requires “each” command processing engine to be capable of processing both pixel and vertex command threads, as ATI contends. The claims require only that each command processing engine in claim 5 be capable of receiving and processing the thread it receives. Ex. 2005, 988:8-19. For example, the system may be configured such that certain command processing engines in the

plurality only process vertex command threads, while others only process pixel command threads. Nothing in the claims or ATI's cited Federal Circuit decision (*ATI Techs. ULC v. Iancu*) requires anything different. *Accord*, Ex. 1011, 8-12.

III. GROUND 1: STUTTARD IN VIEW OF THE KNOWLEDGE OF A POSITA

The Board has already determined that claims 1, 2, and 5-7 of the '053 patent are obvious in light of Stuttard and the Admitted Prior Art. Pet. 4-5. *Accord*, Ex. 1011, 92-102. In doing so, the Board considered and rejected essentially all the arguments ATI asserts.

A. “At The Same Time” Is Not A Claim Limitation

The claims do not require storing both pluralities of command threads at the same time. *See* Section II.A-C. At most, the claims would only require the memory to be “capable of” storing both at the same time. Dr. Pedram testified that would be obvious. Ex. 1004, ¶¶63-67. In contrast, Dr. Hart admitted that he never formed an opinion of whether Stuttard's memory was capable of storing both pluralities at the same time. Ex. 1016, 58:12-60:4. Dr. Hart also admitted that he did not consider what the typical size of a cache memory is, or how many instructions would be in typical command thread, or what size would be needed to save the typical instruction. *Id.*, 69:21-75:14. Dr. Hart's opinion is unsupported and lacks any credible analysis.

Regardless, Stuttard's memory stores both types of command threads at the same time and no modification is necessary. Pet. 20-25; Ex. 1004, ¶¶50-54, 62. Even if modification were needed, it is trivial and Stuttard and the Admitted Prior Art render this limitation obvious. Pet. 22. As the Board noted, Stuttard has a segmented cache that shows first and second portions for storing different types of command threads at the same time or, at a minimum, would render this limitation obvious, as was a well-known practice in the art. Ex. 1005, 86:24-87:10, claim 103; Pet. 22-25; Ex. 1001, 1:42-46, Fig. 1; Ex. 1004, ¶¶62, 63, 67.

This limitation is also rendered obvious by the Admitted Prior Art. Ex. 1001, 1:1-2:9, Figure 1. Dr. Hart admitted that the FIFO buffers 104 and 106, as shown in prior art Figure 1, constituted memory, each storing different types of a plurality of command threads at the same time. Ex. 1016, 14:1-20, 15:4-14, 16:20-17:3. Dr. Pedram testified it would have been obvious that the different types were pixel and vertex command threads. Ex. 1004, ¶64. Dr. Hart admitted that he never formed an opinion whether or not FIFO buffers 104 and 106 are two portions of the same memory or are two separate memories. Ex. 1016, 21:7-21. Dr. Pedram's opinion is un rebutted. Ex. 1004, ¶64. And ATI's arguments to the contrary are incorrect, as shown below.

1. Stuttard Is Not Limited to Phased Operation

ATI argues that Stuttard can only operate in a phased manner, where pixel and vertex operations never run at the same time, so there is no reason to assume that both pixel and vertex command threads are stored at the same time. Their argument has two flaws. First, as recognized by the Board in IPR2015-00325 and the Board in this action, Stuttard is not limited to phased operations. ATI ignores the overall and fundamental teachings of Stuttard that dividing its processing elements (PEs) into a plurality of processing blocks creates advantages by allowing simultaneous processing of different types of tasks. Ex. 1004, ¶¶51-53; Ex. 2005, 995:16-996:5; Ex. 1005, 56:3-9, 82:14-18. Instead, ATI is focused narrowly on its view of how each processing block works, but fails to consider Stuttard as a whole, which can have different tasks running on different processing blocks via its thread manager and scheduler. Ex. 1011, 98-100. Stuttard expressly states this in explaining that “**at any particular moment in time** each thread may be assigned to one of any number of tasks” including 2D, 3D, and general system control type tasks. Ex. 1005, 11:19-25. This expressly discloses that more than one task (of different types) may be running at any particular moment in time. While it is true that one must process vertex threads before a pixel thread can be generated, this does not negate the fact that Stuttard, taken as a whole, renders obvious simultaneous processing of both

pixel and vertex operations, as different PEs can be performing different tasks. Ex. 2005, 40:21-25; Ex. 2008, 166:24-168:11.

ATI's binning arguments ignore the fact that each processing block has its own binning unit, which renders all their arguments void. Ex. 1016, 66:21-67:1 (Dr. Hart admitted that each processing block has its own binning unit.). Even if each processing block were "phased," each processing block can be operating on different types of threads at the same time. Ex. 1005, 11:16-25.

Further, ATI's argument regarding pixel processing being more resource intensive ignores obvious and common real-world considerations. For example, toward the end of a pixel processing task for a frame, some of the processing blocks may be idle due to lack of any pixel command threads. These idle processing blocks can be used to start vertex processing of the next frame. Alternatively, consider a video frame which sees an update on only two regions; one can be in pixel processing because its vertex processing was done quickly, while the other is still in vertex processing because the region image was more complex. These examples illustrate why it would have been obvious that Stuttard's system may process vertices and pixels at the same time.

2. Stuttard Operates on More Than One Instruction at a Time

ATI's argument that Stuttard is limited to operating on one instruction at a time is premised on the fatal flaw that Stuttard is limited to a "single SIMD" array.

As Dr. Pedram testified, Stuttard discloses “multiple SIMD processing blocks” where “each processing block is able to process respective groups of data items.” Ex. 1004, 51-53; Ex. 2005, 995:16-996:5. In light of Stuttard’s additional disclosures as discussed herein, a POSITA would understand that each processing block can process a different instruction stream. *Id.*; Ex. 1005, 11:16-25; Ex. 2010, 238:12-239:13.

For example, Stuttard repeatedly discloses its desire to keep the processing blocks (and not thread processors) as active as possible. Ex. 1005, 11:16-35 (“The thread processors 1026 control the issuance of core instructions from the thread manager so as to maintain processing of simultaneously active program threads, *so that each of the processing blocks 106 can be active for as much time as possible.*”) ATI’s argument that this statement is directed to the thread processors and not the processing blocks is wrong. If the goal was only to prepare a next thread to run just in case the current thread stalls, only one other thread processor would be needed, not seven more. Ex. 1015 ¶13.

ATI’s argument that processing blocks operate on different data, not different instructions is also wrong. Dr. Pedram testified that each of Stuttard’s processing blocks can run its own distinct thread independently of the other blocks, i.e., execute different tasks. Ex. 1004, 51-53; Ex. 2005, 995:16-996:5. ATI’s argument that the Stuttard’s division of PEs is solely to allow processing of different regions of the

display while operating on the same instructions again is focused on a single processing block. Stuttard has multiple processing blocks that can each operate on different instructions. *Id.* Moreover, there is no teaching in Stuttard that vertex processing for the full image must be done before any pixel processing can begin.

ATI is wrong that Stuttard's multiple processing blocks are just for redundancy. First, there is express disclosure in Stuttard that having multiple smaller processing blocks as opposed to just one large processing block improves utilization of the PEs (see Ex. 1005, 2:1-13) and enables execution of multiple threads in parallel as explained above. Second, Stuttard's teaching about redundancy and fault tolerant design is to add one additional redundant PE for each panel of 32 PEs. *Id.*, 9:23-10:3. With this redundancy scheme, there is no advantage to having multiple parallel smaller blocks, because each block, whether large or small, can be broken down into panels of 32 PEs with one extra redundant PE per panel.

Finally, ATI's attempt to construe Stuttard's claims 103 and 104 misses the mark and cannot override the overall teachings of Stuttard. Claim 103 supports that each processing block is designed to handle data items based on the instruction items it receives, which allows for data items to be processed in a plurality of unique ways given the plurality of processing blocks operating on separate instruction items. Ex. 1015 ¶16. Claim 104 supports that the processing elements within each processing block can determine the data's address stored in local memory by referencing the

instruction items associated with that specific block. *Id.* Claims 103 and 104 only show that with a plurality of distinct processing blocks working on a plurality of different instruction items, Stuttard's system allows for data items to be processed in a plurality of unique ways.

3. Stuttard Need Not Care About Command Type to Run Multiple Types

ATI's argument that Stuttard cannot run multiple types of command threads because it does not care about command type is based on the wrong premise that only one thread can be active in Stuttard's system (*i.e.*, phased system). With multiple threads active, even if the selection of the set of active threads is done independently of their type, multiple threads of different types can be activated by the scheduler 1025 and launched by the thread processors 1026 to run concurrently on the parallel processing blocks.

4. “Operative to” Requires Only “Capability”

In the '053 patent, the claim term “operative to” only requires capability. *Aspex* explains that the construction of “operative to” is case specific and, in that particular case, the patent was clearly using the narrower meaning. That is not the case here. As adopted in the ITC, “operable to” or “operative to” as recited in the '053 patent claims simply means “capable of.” Ex. 2002, 79 (citing *In re Imes*, 778 F.3d 1250, 1253-54 (Fed. Cir. 2015); *ParkerVision, Inc., v. Qualcomm Inc.*, 903 F.3d 1354, 1361 (Fed. Cir. 2018)). Regardless, Petitioner has also shown that a POSITA

would have understood and/or found it obvious that Stuttard's system is “configured to” by virtue of at least the following facts: 1) Stuttard’s SIMD array of PEs is partitioned into a plurality of (e.g., eight) different parallel processing blocks; 2) there are a plurality of (e.g., eight) different thread processors orchestrating the flow control of a plurality of (e.g., up to eight) different instruction streams into the plurality of (e.g., eight) processing blocks. *See, e.g.*, Ex. 1005, Figure 4.

B. Stuttard Discloses And Renders Obvious Storing “Complete” Command Threads Even Though Not Required by the Claims

ATI’s arguments are based on its incorrect claim construction requiring the storage of “complete” command threads. *See* Section II.A. Regardless, as the Board noted, Stuttard never says it is storing “partial” command threads. ID, 29-30; *see also* Section II.A. Moreover, it is at least obvious that Stuttard’s memory is capable of storing complete command threads. Ex. 1015 ¶20.

ATI alleges that Dr. Pedram’s testimony was contradictory regarding what Stuttard’s arbiter must know. POR, 33. ATI provides no cites to the alleged testimony at issue, preventing Petitioner from responding. However, a POSITA would have understood that Stuttard’s arbiter must know the type of the thread it is selecting, even if its selection is not dependent on the type of the thread. Ex. 1015 ¶20.

ATI characterizations of Dr. Pedram's testimony is wrong. Dr. Pedram did not take the position that Stuttard's system is limited to prefetch and execute operations as suggested so or that Stuttard cannot store complete command threads.

C. Stuttard Renders Obvious the Arbiter Claim Limitation

ATI argues that Stuttard does not render obvious the claimed arbiter because it cannot select between a pool of two pixel and two vertex command threads. This is based on two flawed premises: that the claims require the presence of both pluralities at the same time, and that Stuttard is limited to a phased system. Both premises are incorrect. *See* Sections II.A.-C. and III.A.1. Because Stuttard can process both pixel and vertex operations at the same time, both are stored at the same time. Ex. 1015 ¶23. The Admitted Prior Art also renders this selection obvious. Ex. 1001, 1:32-64, Figure 1; Ex. 1011, 99. At a minimum, storing both pluralities at the same time and selecting between both pixel and vertex command threads is obvious. *Id.* The Board agreed and rejected essentially the same arguments by ATI in IPR2015-00325. Ex. 1011, 98-100.

Further, Stuttard discloses selecting based on the relative priorities of the pixel and vertex command threads. Ex. 1005 5:56-59 ("The Threads are assigned priorities relative to one another."); 10:49-12:54; Ex. 1011, 99 (Stuttard and APA disclose limitation). ATI's additional arguments are equally unpersuasive.

Dr. Hart admitted the Admitted Prior Art in the '053 patent discloses an arbiter capable of selecting between a plurality of pixel command threads and a plurality of vertex command threads, *e.g.*, “arbiter 102 as shown in Figure 1 can select command threads from either FIFO buffer memory 104 or FIFO buffer memory 106.” *Id.* 13:8-20. Dr. Hart further admitted (and Figure 1 shows) that arbiter 102 can select from the plurality of command threads in buffer 104 and the plurality of command threads in buffer 106, which are both present at the same time, even if it must select the first out command thread from either buffer. Ex. 1016, 16:4-9.

Finally, ATI is also wrong that the APA does not disclose types of command threads. A POSITA would understand from the '053 patent's background section that there existed two types of threads in the admitted prior art: a first type requiring ALU resource (corresponding to the vertex command thread) and another type requiring the texture fetch resource (corresponding to the pixel command thread.) Ex. 1001, Figure 1, 1:38-40. Also, as shown in Figure 1 (the APA) of the '053 patent, arbiter 102 looks to the reservation stations (FIFO buffers) 104 and 106 and chooses either a vertex command thread (from 104) or a pixel command thread (from 106), to be provided to a texture fetch processor. Ex. 1001, 1:42-48. Thus, the APA in the '053 patent explicitly teaches that the arbiter can choose one thread from among two types of command threads.

D. **Stuttard Renders Obvious A Texture Processing Engine**

Dr. Pedram testified that the *specific disclosures of Stuttard* regarding lighting and shading operations would lead a person to understand that a texture processing engine was necessary. Ex. 1004, ¶84; Ex. 2008, 264:23-265:8 (“a POSITA looking *at this disclosure in the context of Stuttard’s reference* would understand this lighting and shading to be examples of the texture processing engine.”) ATI’s only argument is that “lighting and shading operation” *in general* do not *necessarily* require a texture processing engine. Ex. 2009 ¶97 (“Some lighting and shading processes only require vertex processing, and thus do not require a texture processing engine.”). ATI fails to address the specific disclosures of Stuttard. Even if true that not all lighting and shading necessarily requires a texture processing engine, one obvious way to do lighting and shading is still to use a texture processing engine, as testified by Dr. Pedram. Ex. 1004 ¶84; Ex. 1015 ¶24.

IV. **GROUND 2: STUTTARD IN VIEW OF WILLIAMS AND THE KNOWLEDGE OF A POSITA**

Williams is further evidence of what was known in the art (within the knowledge of a POSITA) at the time of the invention. Pet. 40. To the extent not considered shown and/or obvious by Stuttard alone, the Petition shows that a POSITA would have been motivated to use Williams’ teachings regarding storing pixel and vertex command threads in a first and second portion of memory, respectively, when practicing Stuttard. Pet. 40-50.

ATI's two primary arguments are that Williams does not store vertex and pixel command threads in first and second portions of memory and that Williams does not store command threads at all. Both are wrong as explained below.

A. Storage of command threads in first and second portions of memory

The Petition shows that because Williams transfers different types of graphics commands across different buses, it would have been obvious to store different types of commands in different portions of memory. Pet., 42. This would result in increased benefits, including using consecutive addresses with fewer instructions, resulting in increased efficiency, which is a goal of Williams. *Id.*

ATI's rebuttal is that there is no benefit to storing the command threads in different portions because the GBS already performs an equivalent function and the new system would be less efficient and slow. What ATI omits is that if the GBS knows that all vertex command threads are in one portion of the memory whereas the pixel command threads are in the other portion, it will simply go to the appropriate portion of the memory and move the threads from there to the appropriate buses. Ex. 1015 ¶26. This design will streamline the memory access and thread assignment to buses. *Id.* Further, the GBS sorts the commands as it receives them before transmitting them to a bus interface. *See, e.g.*, Ex. 1006, 18:14-

26. That would still take place even with the combined system. That is, the GBS functionality has no impact on the efficiency created by storing in different portions.

ATI's suggestion that commands come out of the storage buffer in the same order they went in is unsupported. There is no disclosure that the commands stored in the buffer are retrieved in the same order they went in. Regardless, how the graphics commands are stored in the system memory 108 will materially impact efficiency. In particular, a storage scheme in which accessing commands of a particular command type does not require "address jumps" would be beneficial. Therefore, the system memory 108 may be partitioned into two portions, and subsequently, there will be a benefit on the "read" end, when the commands are coming out of the system memory 108. Ex. 1004, ¶¶97-99.

ATI alleges that additional components would be needed for the combination, which would nullify any benefits. ATI omits that this can be done with tagging that is already disclosed in Williams. Williams discloses that it "tags the multiple graphics commands (for example, graphics API commands) that are stored within input buffer 202 to thereby indicate the specific order of the commands" before it "assigns each of the multiple graphics commands to one of the plurality of buses 116A-116N, which transfer the multiple graphics commands between GBS 204 and GBD 210." Ex. 1006, 11:25-12:2. Adding the additional detail in the tagging is obvious and not cost prohibitive. Ex. 1015 ¶27.

Petitioner relies on Stuttard for the claimed arbiter. Pet. 46-50; Ex. 2010, 282:1-7. Thus, ATI's complaint about how the arbiter would work in Williams is a problem that does not exist. Likewise, ATI's statement that the GBS is not coupled to the output buffer appears to suggest ATI is requiring a direct connection, which is not required and has no effect on the claimed combination. Ex. 1015 ¶28.

Finally, ATI argues that there is no benefit to adding a GBS to identify command type in Stuttard. This ignores Petitioner's primary point, that Stuttard would benefit from partitioning the memory into different portions for each command type, as suggested and made obvious by Williams.

B. Stuttard Stores "command threads"

The Petition shows that Stuttard stores command threads. Moreover, Williams' OpenGL commands are high level commands that give rise to vertex or pixel processing. Ex. 1015 ¶30. Further, as the Board noted, the accumulation of commands in display list 704 discloses complete command threads. Ex. 1006, 24:16-17. Thus, the storing of pluralities of pixel and vertex command threads is, at least, obvious based on the disclosures of Stuttard and/or Williams.

ATI's argument that the display list 704 cannot be the claimed memory device because of its location relative to the GBS is a strawman. The claimed arbiter for this ground is in Stuttard, not Williams. Therefore, the relative locations of Williams memory to its GBS are irrelevant.

C. Stuttard Satisfies the Claimed Arbiter

ATI fails to identify any issue that Stuttard's arbiter may encounter or any modification that may be required. Stuttard's arbiter needs no substantive modification simply because the memory stores the command threads, which it is already doing, in certain portions of the memory based on Williams.

D. A Second Memory is an Obvious Design Choice Within the Knowledge of Skill in the Art, Which Williams Supports

Dr. Pedram testified that using multiple memory devices for the two portions, rather than a single device, such as Stuttard's memory cache, is nothing more than a simple design choice. Ex. 1004, ¶111. Further, the concept of storing different types of threads in different memories (a first and second memory) is evident in the knowledge of skill in the art, such as in Williams, which stores different types of threads in different memories. *Id.*, 110-111.

E. Motivation to Combine and Reasonable Expectation of Success

Dr. Pedram walks through a host of benefits for combining Stuttard and Williams and explains that both are directed to the same goal of increasing the speed of graphics processing. Ex. 1004, ¶¶101-106. ATI attacks the substantial motivations to combine and reasons for success presented by Dr. Pedram, but none are persuasive.

There is no support for ATI's argument that Williams' memory would be slower than Stuttard's cache. Regardless, Petitioner's ground is that Stuttard's cache

would be partitioned based on the teachings of Williams, not that the cache would be replaced with a different type of memory. ATI argues that Williams' objective is not increased speed, but increased bandwidth to transfer data. Even if true, the concepts are related. System performance (speed) will increase with more computing power and with higher bandwidth. Moreover, it is a red herring to argue that the bandwidth from the host to the graphics system is a limiting factor in Stuttard. A POSITA would recognize that graphics processing is a highly data intensive task and in many cases the performance bottleneck can be set by the data transfer speed. Ex. 1015 ¶34.

ATI next argues that Stuttard does not have a problem with AGP busses and would not need the buses of Williams. Again, Petitioner does not require that Williams' dedicated bus architecture be incorporated into Stuttard. However, to increase the computational power of Stuttard's system, more parallel blocks can be added (*e.g.*, Stuttard states that 8 parallel blocks are just an example design.). With more blocks, data transfer rates become an even more important issue; and hence a POSITA would be motivated to combine Stuttard and Williams to increase the system performance.

Finally, ATI creates another strawman to argue that Petitioner's reasonable expectation of success cannot be correct because it is based on "generic reasons" that would apply in every situation. First, both references are directed to similar

problems in the same field. And both are using “generic” components that would have predictable results and require routine modification without undue experimentation. Ex. 1015 ¶35. The fact that they also share some of those common components adds to the confidence of any combination. Second, ATI’s complaint for “lack of details” is disingenuous. Using two memories instead of a single memory or storing different types of command threads in different portions of the same memory is nothing more than a design choice that is trivial at best. Ex. 1004, ¶111.

V. GROUND 3: WILLIAMS IN VIEW OF WHITTAKER AND THE KNOWLEDGE OF A POSITA

A. Storage of Pixel or Vertex Command Threads

Contrary to ATI’s position, Williams renders obvious storing command threads. *See* Section IV.B. Williams discloses that groups of graphics commands are ordered within a sequence of graphics commands. Ex. 1006, 3:1-14, 14:8-21. Dr. Pedram testified that a POSITA would have understood each sequence of graphics commands constitutes a command thread. Pet. 46-47, 51, 56; Ex. 1004, ¶ 88.

B. Williams Discloses or Renders Obvious the Claimed Arbiter

The Petition identifies that Williams’ GBS is the claimed arbiter. Pet. 56-57; Ex. 1004, ¶133. ATI argues that the GBS does not *select* graphics commands because it only *receives* them and *sorts* them. ATI contends that this cannot be

“selecting” because it is not done from memory and there is no “picking out.” ATI’s arguments ignore the evidence and add limitations to the claims.

As the Petition shows and Dr. Pedram explains, GBS *sorts* and *assigns* graphics commands to a plurality of busses according to the type of command, *i.e.*, based on whether it is a geometry/vertex command or a pixel command. *Id.* The sorting and assignment of the commands is the claimed “selection.” Further, The claims do not require the selection be made “from memory,” such that the arbiter must make a selection before it receives inputs. The claim only requires the arbiter be coupled to the memory and that it make a selection from the plurality of command threads. Williams discloses sorting, assigning, and placing commands on particular busses based on resource availability, which is the claimed “selecting.” Ex. 1004, ¶133; Ex. 1006, 8:20-24, 12:11-14; Ex. 2010, 197:12-22.

ATI argues that Williams’ “sorting” cannot be a “selection.” Untrue. Commands are stored in the input buffer; they are then analyzed, tagged, and moved to different buses. A POSITA would understand this to be an act of arbitration among commands, *i.e.*, a selection. Ex. 1014 ¶39. Williams expressly discloses that its arbiter can select command threads (*i.e.*, sort and assign commands to the plurality of buses) based on different “assignment schemes,” for example, “according to a type of command (e.g., geometry, texture)” or “based on the availability of a bus, to balance bus load, to evenly distribute commands, or to

unevenly distribute traffic over the busses 116A-116N.” Ex. 1006, 8:20-24, 12:11-14.² *See also* Ex. 1006, 8:14-19 (tagging of commands is performed before the commands come out of the input buffer).

To the extent “selection” is not considered disclosed in Williams, Whittaker discloses “selecting threads based on relative priorities.” Ex. 1004, ¶¶134-137. Whittaker’s MCC2 performs the priority thread selection. *Id.*; Ex. 1007, 3:28-30. At a minimum, the combination of Williams and Whittaker render the claimed arbiter obvious.

C. Claims 2, 5-9 – Williams’ Command Processing Engines Process the Command Thread They Receive from the Claimed Arbiter

ATI argues that Williams’ command processing engines (402 and 404) cannot satisfy the claim because they can only process pixel command threads or vertex command threads but not both. Nothing in claims 2 or 5-9 requires the command processing engines to process both pixel and vertex command threads. *See* Section II.D.

² ATI again mischaracterizes Dr. Pedram’s testimony to argue that Williams’ arbiter does not “select” if it merely receives inputs. Dr. Pedram was answering a hypothetical question not directed to Williams. Ex. 2010, 205:17-20. Dr. Pedram also testified that a “selection” may include assigning and placing commands on busses. *Id.*, 197:12-22.

D. Claim 6 – Williams’ Disclosure Renders Claim 6 Obvious

ATI states that the Petition cites only to portions of Williams that discloses various calculations such as “geometry calculations, curve and surface geometry, processing geometric primitives, and operations on pixels.” However, the Petition also cites to the testimony of Dr. Pedram that explains those calculations would “necessarily include an arithmetic logic unit to perform at least the necessary geometry, or vertex, operations.” Ex. 1004 ¶148.

E. Claim 7 – Williams Renders Obvious a Texture Processing Engine

The Petition asserts that claim 7 is obvious based on the disclosures of Williams and Whittaker. Pet. 51, 56-61 (citing Williams, 3:1-3 and Dr. Pedram’s declaration ¶149). Williams discloses that its command processing engines (components 402 and 404) “can be functional components of graphics subsystem 102 that perform geometry and pixel operations, respectively.” Pet. Ex. 1004, ¶149. Williams further discloses that the graphics subsystem 102 “processes texture data in order to create lifelike surfaces and shadows within a 3D image.” *Id.*; Pet. 61. A POSITA would understand these disclosures “necessarily include a texture processing engine.” Ex. 1004, ¶149.

F. Motivation to combine and reasonable expectation of success

Each of ATI’s attacks on Dr. Pedram’s explanations for motivations to combine and a reasonable expectation of success are without merit. *See, e.g.*, Ex. 1004, ¶¶138-139.

ATI's argument that the teachings cannot be combined because they have different approaches (Williams maintains order, while Whittaker assigns priorities) is nonsense. It is often the case that a POSITA will combine a different approach to a common problem to achieve an obvious result. If the approaches were the same there would be no point to combining them. Moreover, as discussed above, nothing in Williams requires the order to be maintained. *See* Section IV.A.

Williams and Whittaker are both directed to 3D graphics processing and rendering. Ex. 1004, ¶138. Dr. Pedram has testified that both are closely related data processing types. *Id.* In fact, both have arbiters that carry out thread selection in multi-threaded processing systems. *Id.* Both have similar problems, *e.g.*, resource availability and a desire for higher performance. *Id.* Both have common goals of improving efficiency related to the transfer of information and commands in graphics processing systems. *Id.* ¶139. These are more than sufficient to find a motivation to combine.

ATI's attempt to create a conflict between the two references is without merit. A POSITA reading the references would know that while Williams' system can be used to increase the data transfer bandwidth to a graphics subsystem, Whittaker's teachings can be applied to allow Williams' system to achieve its goal while also honoring command priorities. Ex. 1014 ¶47.

Dated: April 10, 2024

Respectfully submitted,

/Jeffrey L. Johnson/

Jeffrey L. Johnson
Reg. No. 53,078
BAKER BOTTS, LLP
910 Louisiana Street
Houston, TX 77002-4995
Telephone: 713-229-1234
Facsimile: 713-229-1522
Email: Jeffrey.Johnson@bakerbotts.com
DLBBRealTEKIPRs@bakerbotts.com

Christopher J. Higgins
Reg. No. 66,422
ORRICK, HERRINGTON & SUTCLIFFE LLP
Columbia Center
2100 Pennsylvania Avenue NW
Washington, D.C. 20037
Telephone: (202) 339-8400
Facsimile: (202) 339-8500
Email: 0CHPTABDocket@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

*Counsel for Petitioner Realtek
Semiconductor Corp.*

CERTIFICATE OF SERVICE

In accordance with 37 C.F.R. § 42.6(e), the undersigned certifies that on April 10, 2024, a complete and entire copy of this **PETITIONER'S REPLY TO PATENT OWNER'S RESPONSE TO PETITION** and Exhibits 1015-1016 were served on Patent Owner via electronic mail at the following addresses:

William A. Meunier
wameunier@mintz.com

Michael T. Renaud
mtrenaud@mintz.com

Adam S. Rizk
arizk@mintz.com

Reza Dokhanchy
rdokhanchy@mintz.com

AMD_IPRS@mintz.com

/Jeffrey L. Johnson/
Jeffrey L. Johnson (Reg. No. 53,078)
Lead Counsel for Petitioner

**CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME
LIMITATION, TYPEFACE REQUIREMENTS, AND TYPE STYLE
REQUIREMENTS**

1. This **Petitioner's Reply to Patent Owner's Response** complies with the type-volume limitation of 5,600 words, comprising 5593 words, excluding the parts exempted by 37 C.F.R. § 42.24(a).

2. This **Petitioner's Reply to Patent Owner's Response** complies with the general format requirements of 37 C.F.R. § 42.6(a) and has been prepared using Microsoft® Word in 14-point Times New Roman.

Respectfully submitted,

BAKER BOTTS L.L.P.

/Jeffrey L. Johnson/
Jeffrey L. Johnson (Reg. No. 53,078)
Lead Counsel for Petitioner

Date: April 10, 2024