

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

**DECLARATION OF DR. MASSOUD PEDRAM IN SUPPORT OF
PETITIONER'S REPLY TO PATENT OWNER'S
RESPONSE TO PETITION**

Table of Contents

	Page
I. INTRODUCTION.....	4
II. CLAIM CONSTRUCTION	4
A. “command threads”	4
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”	4
C. “select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads”	5
D. “command processing engine”	5
III. GROUND 1: STUTTARD IN VIEW OF THE KNOWLEDGE OF A POSITA.....	6
A. “At The Same Time” Is Not A Claim Limitation	6
1. Stuttard Is Not Limited to Phased Operation	7
2. Stuttard Operates on More Than One Instruction at a Time	9
3. Stuttard Need Not Care About Command Type to Run Multiple Types	11
4. “Operative to” Requires Only “Capability”	11
B. Stuttard Discloses And Renders Obvious Storing “Complete” Command Threads Even Though Not Required by the Claims	12
C. Stuttard Renders Obvious the Arbiter Claim Limitation	14
D. Stuttard Renders Obvious A Texture Processing Engine	15
IV. GROUND 2: STUTTARD IN VIEW OF WILLIAMS AND THE KNOWLEDGE OF A POSITA	16
A. Storage of command threads in first and second portions of memory.....	16
B. Stuttard Stores “command threads”	18
C. Stuttard Satisfies the Claimed Arbiter.....	19
D. A Second Memory is an Obvious Design Choice Within the Knowledge of Skill in the Art, Which Williams Supports	19
E. Motivation to Combine and Reasonable Expectation of Success.....	19

V. GROUND 3: WILLIAMS IN VIEW OF WHITTAKER AND THE KNOWLEDGE OF A POSITA	21
A. Storage of Pixel or Vertex Command Threads	22
B. Williams Discloses or Renders Obvious the Claimed Arbiter.....	22
C. Claims 2, 5-9 – Williams’ Command Processing Engines Process the Command Thread They Receive from the Claimed Arbiter.....	24
D. Claim 6 – Williams’ Disclosure Renders Claim 6 Obvious	24
E. Claim 7 – Williams Renders Obvious a Texture Processing Engine.....	24
F. Motivation to combine and reasonable expectation of success	24

I, Dr. Massoud Pedram, declare as follows:

I. INTRODUCTION

1. This is my Reply Declaration in Support of the Petition (“Reply Declaration”). My Reply Declaration responds to the technical issues raised in Patent Owner Response and Dr. Hart’s associated declaration. Nothing in Patent Owner’s Response has changed any of my opinions that the challenged claims are invalid.

II. CLAIM CONSTRUCTION

A. “command threads”

2. I have testified that 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. *See, e.g.*, Ex. 2008, 183:6-10. Even if the claim requires “complete” command threads, with which I disagree, a POSITA would have understood that storing/receiving/executing the sequence of instructions for a command thread were stored over time would satisfy the claim.

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”

3. In my opinion, this claim element does not require storage of both types of command threads “at the same time.” While the memory must be capable of storing both pluralities of command threads, the plain language of the claim does not

require that both pluralities need be present “at the same time.” Nothing in the arbiter limitation requires such a reading as discussed below.

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

4. A POSITA would not understand this claim limitation to be limited to a pool of two pixel and two vertex command threads to be present at the time of selection. While that may be within its plain language, it is equally likely that there may be a plurality of pixel command threads in a first memory device, and a separate plurality of vertex command threads in a second memory device, as recited in claim 3, where the arbiter may simply select from either plurality without “considering” the other. The claim requires only that the arbiter select from either bucket one or bucket two provided each bucket as a plurality of pixel or vertex command threads.

D. “command processing engine”

5. In my opinion, 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.

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

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

6. As explained above, storing a plurality of pixel and a plurality of vertex command threads “at the same time” is not a claim requirement. There does not appear to be any dispute that both are stored in Stuttard’s cache memory. Only that they are not stored at the same time. However, Stuttard’s parallel processing blocks allow for simultaneous execution of pixel and vertex command threads. Thus, as I have testified, a POSITA would have found it obvious that Stuttard is storing both pluralities at the same time. Ex. 1004, ¶¶63-67. In contrast, Dr. Hart admitted that he never formed any opinions related to whether Stuttard’s memory was capable of storing both pluralities at the same time. Ex. Depo, 55:18-57:11. The reason is because it is beyond dispute.

7. 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.

8. This limitation is also rendered obvious by the Admitted Prior Art. Ex. 1001, 1:22-2:9, 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. Depo, 11:2-21, 12:5-15, 13:21-14:4. 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. Depo, 18:8-22. Dr. Pedram's opinion is unrebutted. Ex. 1004, ¶64.

1. Stuttard Is Not Limited to Phased Operation

9. Stuttard's parallel processing blocks are designed to execute multiple tasks at the same time. Thus, it is my opinion that Dr. Hart's narrow reading that Stuttard can only operate in a phased manner, where pixel and vertex operations never run at the same time, is wrong. Dr. Hart 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. Dr. Hart's tunnel vision looking solely at what happens within a single processing block is too restrictive. Further, his reliance on the binning processes to limit all processing blocks ignores the fact that each processing block has its own binning unit. 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.

10. Further, it is directly contradictory to Stuttard’s express disclosure 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. Ex. 2005, 40:21-25; Ex. 2008, 166:24-168:11.

11. 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

12. Dr. Hart's tunnel vision carries over to other parts of his analysis. For example, he limits all of Stuttard's processing blocks to a single SIMD array, but 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. Taking Stuttard's entire disclosure as a whole, 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.

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.

14. Dr. Hart's argument that processing blocks operate on different data, not different instructions is also wrong. 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. In my opinion, the 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 and ignores the overall objectives of Stuttard. There is no teaching in Stuttard that vertex processing for the full image must be done before any pixel processing can begin.

15. Dr. Hart 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.

16. Finally, ATI and Dr. Hart'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. 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. 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

17. It is beyond dispute that Stuttard has multiple threads active. Even if the selection of the set of active threads is done independently of their type, a POSITA would have understood that 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”

18. I understand that operative to means only 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)).

19. Regardless, a POSITA would have understood and/or found it obvious that Stuttard's system satisfies the claim limitation even under Dr. Hart's more restrictive standard of “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

20. As I have explained, Stuttard never says it is storing “partial” command threads. ID, 29-30; *see also* Section II.A. In fact, a POSITA would have considered Stuttard to disclose storing complete command threads. Stuttard’s system is designed for parallel processing, where multiple threads can execute simultaneously. Storing complete instructions for each thread allows the system to access and execute them independently and in parallel, thereby maximizing processing efficiency. By storing all instructions for a thread in contiguous memory locations, Stuttard’s system can minimize the latency associated with fetching individual instructions from different memory locations. This can speed up instruction fetching and processing, improving overall performance. Storing complete thread instructions in the memory allows for simpler control flow, as Stuttard’s system can easily switch between different threads without needing to retrieve additional instructions from other memory locations. Storing thread instructions together can also enhance data locality by ensuring that related data accessed by a thread is located close together in memory. This can reduce memory access times and

improve overall data processing speeds. At a minimum, a POSITA would have found it obvious that Stuttard's memory is capable of storing complete command threads. Further, 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. More precisely, Stuttard's arbiter is coupled to the Stuttard's memory device, which is configured to store vertex and pixel command threads in two different portions of the memory. Therefore, Stuttard's arbiter, which is operable to select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads, must know the type of thread it is selecting, although the selection itself is based on relative priorities of the plurality of pixel command threads and the plurality of vertex command threads.

21. I never took the position that Stuttard's system is limited to prefetch and execute operations as ATI suggests or that Stuttard cannot store complete command threads. ATI tries to link my generic testimony about pre-fetch and execute operations of typical systems to Stuttard. They are mistaken. The testimony speaks for itself, e.g., "I just need to provide a little bit of more technical context and background ...", "Standard processing is ...", "That's typically what happens in these kinds of systems." I was merely testifying that it is possible in some systems to overlap the fetch and execution of instructions in a thread. Thus, while the thread's

instructions are being received, some of the already received instructions are sent to the command processing engine to be executed.

C. Stuttard Renders Obvious the Arbiter Claim Limitation

22. As discussed in the claim construction section, it is my opinion that Dr. Hart is interpreting the “arbiter selection” claim limitation too narrowly. *See* Sections II.C. In addition, Dr. Hart continues in his overly restrictive view of Stuttard as limited to a phased system. *See* Section III.A.1-3.

23. In reality, Stuttard can process both pixel and vertex operations at the same time. Consequently, it is at least obvious that two types of command threads are stored at the same time as was well-known to do at the time of the invention. Ex. 1001, 1:32-64, Figure 1; Ex. 1011, 99. Further, I disagree that this APA in the '053 patent 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, a POSITA would have understood that 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 pluralities of two types of command threads, pixel and vertex. At a minimum, storing both pluralities at the same time and selecting between both pixel and vertex command threads is obvious. *Id.*

D. Stuttard Renders Obvious A Texture Processing Engine

24. Dr. Hart's distinctions regarding Stuttard's use of a texture processing engine are devoid of any real-world knowledge of a POSITA. At best, his arguments are a bunch of hand waving. I expressly 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 and Dr. Hart'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.”). Even if that were true, it is a deflection away from the specific disclosures of Stuttard. Moreover, 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. Ex. 1004 ¶84. However, in my opinion the disclosure is much stronger than that –

a texture processing engine would be understood to necessarily be present to execute Stuttard's disclosed operations.

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

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

25. I disagree with ATI and Dr. Hart 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.

26. Williams discloses that its GBS recognizes that all vertex command threads are in one portion of the memory whereas the pixel command threads are in another portion of the memory. The GBS will simply go to the appropriate portion of the memory and move the threads from the stored location to its appropriate bus(es). In doing so, memory access and thread assignment is streamlined improving efficiency and speed. By segregating threads based on their type and storing each thread type in a designated portion of the memory, the system can expedite access to pertinent thread instructions and information, diminish the retrieval time required for threads, and enhance overall system performance. In addition, storing thread instructions together in a smaller memory block (portion) enhances data locality because related data accessed by thread instructions is likely to be located nearby in memory. The GBS also sorts the commands as it receives

them before transmitting them to a bus interface. *See, e.g.*, Ex. 1006, 18:14-26. That function of the GBS would still take place even with the proposed combined system. That is, the GBS functionality has no impact on the efficiency created by storing in different portions. or stated differently, it is not true that the GBS function already being performed is an equivalent function as ATI and Dr. Hart suggest and, thus, the efficiency and speed will go up.

27. In addition, there is nothing in Williams to suggest that the commands stored in the buffer are retrieved in the same order they went in. Even if that were true, storing the command threads in different portions according to type will have a material impact on the overall system efficiency. Such a configuration would not require “address jumps,” giving a quantifiable benefit on the “read” end when the commands are coming out of the system memory 108. Ex. 1004, ¶¶ 97-99. Further, no additional components would be needed. A POSITA would understand that the process can be done with the kind of tagging that is already disclosed in Williams. For example, 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 routine, inexpensive, and obvious.

28. Dr. Hart's argument regarding Williams' arbiter is confusing. I did not rely putting Stuttard's arbiter into Williams, or vice versa. Stuttard has an arbiter that satisfies the claims. 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, in my opinion, is not required by the claims. Moreover, whether the connection is direct or indirect is trivial and has no impact on the system.

29. Finally, Stuttard would benefit from partitioning the memory into different portions for each command type, as suggested and made obvious by Williams. Thus, I disagree that there is no benefit to adding a GBS to identify command type in Stuttard.

B. Stuttard Stores "command threads"

30. As I have repeatedly shown, a POSITA would have understood that Stuttard discloses storing command threads. Moreover, Williams' OpenGL commands are high level commands that give rise to vertex or pixel processing. ATI and Dr. Hart's characterization of Williams is also wrong. 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.

C. Stuttard Satisfies the Claimed Arbiter

31. Stuttard's arbiter needs little to no modification simply because the memory stores the command threads in certain portions of the memory based on Williams. Stuttard is already storing command threads and it is obvious they are being stored in different portions. Adopting the idea of storing in two portions of the memory to achieve additional benefits has zero negative impacts on Stuttard's arbiter and would not be understood to require any meaningful modification, if any. Memory allocation and storage schemes as simple as storing in two portions are trivial and routine to implement.

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

32. 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

33. I have already provided a long list and explanation of the benefits a POSITA would expect from combining Stuttard and Williams. Ex. 1004, ¶¶101-106. Both references are directed to the same goal of increasing the speed of

graphics processing. Ex. 1004, ¶¶101-106. The benefits and similarities all go to the motivation to combine and the reasonable expectation of success.

34. I disagree that Williams' memory would be slower than Stuttard's cache. I see no evidence of that, certainly ATI and Dr. Hart provide nothing that a POSITA in the real world would consider persuasive. In my opinion, it is a meaningless argument because I advocated 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 incorrect 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. ATI next argues that Stuttard does not have a problem with AGP busses and would not need the buses of Williams. Again, that was not a requirement of my argument. However, ATI and Dr. Hart are again incorrect. A POSITA would have understood that 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.

35. Finally, I disagree that there is no reasonable expectation of success. ATI and Dr. Hart take issue with the alleged generic nature of the facts, but in my opinion those facts are just so well understood and known in the art that it is all the more reason to expect success. First, both references are directed to similar problems in the same field. A POSITA would naturally look for systems addressing the same problem and would be more confident and ready to accept the proposed solutions when they are in the same field. The fact that both are using or applying “generic” components and/or concepts would encourage the POSITA to adopt those solutions before more complicated and expensive solutions. Moreover, using known components and concepts has a higher chance of success because the results are more predictable and requires only routine modification, if any, without lots of expensive and time-consuming trial and error. 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

36. Williams discloses that groups of graphics commands are ordered within a sequence of graphics commands. Ex. 1006, 3:1-14, 14:8-21. I 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

37. ATI and Dr. Hart imply two issues. First that sorting is not selecting. Second, that the GBS does nothing beyond sorting that could satisfy the claimed “selection.” Both are untrue.

38. As I have shown, Williams’ 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. Pet. 56-57; Ex. 1004, ¶ 133. In my opinion, the sorting and assignment of the commands is the claimed “selection.” The claims do not require the selection be made “from memory,” such that the arbiter must make a selection before it receives inputs. Instead, the claim only requires the arbiter be coupled to the memory and that it makes 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.

39. With respect to ATI's argument that Williams' "sorting" cannot be a "selection," 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. 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. *See also* Ex. 1006, 8:14-19 (tagging of commands is performed before the commands come out of the input buffer).

40. ATI again mischaracterizes my testimony to argue that Williams' arbiter does not "select" if it merely receives inputs. I was answering a hypothetical question not directed to Williams. Ex. 2010, 205:17-20. Contrary to ATI attack on my testimony, I expressly testified that a "selection" may include assigning and placing commands on busses. *Id.*, 197:12-22.

41. 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

42. I see nothing in claims 2 or 5-9 that requires the command processing engines to process both pixel and vertex command threads. *See* Section II.D. All that is required is that the command processing engines process the command thread they receive from the claimed arbiter, which I have shown.

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

43. My testimony with respect to Ground 3 and claim 6 includes that various geometry calculations disclosed in Williams 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

44. I testified that the texture processing engine recited in claim 7 is obvious based on the disclosures of Williams and Whittaker. Ex. 1004 ¶149. ATI is simply incorrect that I did not explain this element.

F. Motivation to combine and reasonable expectation of success

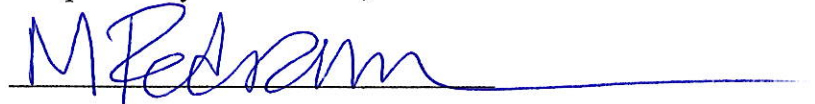
45. As with Ground 2, I provided extensive discussion on the motivation to combine and reasonable expectation of success for the combination of Williams and Whittaker. *See, e.g.,* Ex. 1004, ¶¶ 138-139.

46. 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 in the real world that a POSITA will combine different approaches to a common problem to achieve a result. Indeed, I expect my students to do just that. It is also often the case that when going through that exercise, the result is obvious. The whole point of combining two teachings is that if something is missing in the first approach or if the first approach is not something I can use, I will look to a second approach (something that is different) to fill in the gap or provide an approach that meets my needs. If the two approaches are the same, then why would anyone need to combine them?

47. ATI's attempt to create a conflict between the two references is without merit. A POSITA reading Williams and Whittaker 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.

Dated: April 10, 2024

Respectfully Submitted,



DR. MASSOUD PEDRAM