

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

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

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION	1
II. PROCEDURAL OVERVIEW	4
III. PRIOR IPR PROCEEDING.....	5
IV. TECHNICAL OVERVIEW	7
V. '053 PATENT OVERVIEW	9
VI. PRIOR ART OVERVIEW	12
A. Stuttard (PCT Pub. No. WO00/62182A2)	12
1. Overview	12
2. Phased Architecture	17
B. Williams (PCT Pub. No. WO00/63770A1)	19
C. Whittaker (U.S. Patent No. 5,968,167)	20
VII. Claim Construction.....	20
A. “command threads”	21
B. “at least one memory device . . . operative to store a plurality of pixel command threads and . . . a plurality of vertex command threads”.....	22
VIII. PETITIONER HAS FAILED TO ESTABLISH THE REQUIRED REASONABLE LIKELIHOOD OF SUCCESS FOR ANY GROUND	23
A. Ground 1: Stuttard In View of the Knowledge of a POSITA.....	23
1. Stuttard 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	24
2. Stuttard Does Not Disclose Or Render Obvious Storing Command Threads	33
3. Stuttard Does Not Disclose Or Render Obvious An Arbiter That Selects A Command Thread From The Plurality Of Pixel Command Threads Vertex Command Threads.....	34

B.	Ground 2: Stuttard In View of Williams and the Knowledge of a POSITA	38
1.	Williams 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	38
2.	There Is No Motivation to Combine Williams with Stuttard	45
C.	Ground 3: Williams in View of Whittaker and the Knowledge of a POSITA	48
1.	Ground 3 Is Procedurally and Substantively Incomplete And Cannot Form A Basis For Institution	48
2.	Williams Does Not Disclose or Render Obvious Storing Command Threads	50
3.	Williams Does Not Disclose or Render Obvious an Arbiter That Selects A Command Thread	51
4.	Williams Does Not Disclose or Render Obvious Command Processing Engines	52
5.	Neither Williams Nor Whittaker Discloses or Render Obvious An Arbiter That Selects Command Threads Based On Relative Priorities of the Command Threads	54
6.	There is No Motivation to Combine Williams with Whittaker, Nor Any Reasonable Expectation of Success	56
IX.	CONCLUSION	58

TABLE OF AUTHORITIES

Page(s)

Federal Cases

<i>ATI Techs. ULC v. Iancu</i> , 920 F.3d 1362 (Fed. Cir. 2019)	4
<i>Fidelity Nat’l Info. Servs., Inc. v. DataTreasury Corp.</i> , IPR2014-00489, Paper 9 (Aug. 13, 2014)	50
<i>GN Resound A/S v. Oticon A/S</i> , IPR2015-00103, Paper 13 (June 18, 2015).....	49
<i>Google, Inc. et al. v. EveryMD.com LLC</i> , IPR2014-00347, Paper 9 (May 22, 2014).....	49
<i>Qualcomm Inc. v. Apple Inc.</i> , 24 F.4th 1367 (Fed. Cir. 2022)	32

TABLE OF EXHIBITS

Exhibit	Description
2001	Declaration of Dr. John Hart and CV
2002	Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same, Inv. No. 337-TA-1318, Final Initial Determination (July 7, 2023)
2003	James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. 1990. Computer Graphics: Principles And Practice (2nd ed.). Addison-Wesley Longman Publishing Co., Inc
2004	Akenine-Möller, T., Haines, E., & Hoffman, N. (2008). Real-Time Rendering (3rd ed.). A K Peters/CRC Press.
2005	Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same, Inv. No. 337-TA-1318, Hearing Transcript, Day 4 (March 16, 2023)
2006	Edwin Earl Catmull. 1974. “A Subdivision Algorithm For Computer Display of Curved Surfaces.” Ph.D. Dissertation. The University of Utah
2007	Difference Between Buffering https://techdifferences.com/difference-between-buffering-and-caching-in-os.html
2008	Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same, Inv. No. 337-TA-1318, Deposition of Dr. Pedram (January 04, 2023)

I. INTRODUCTION

Pursuant to 35 U.S.C. § 313 and 37 C.F.R. § 42.107, Patent Owner, ATI Technologies ULC (“ATI” or “Patent Owner”) submits this Preliminary Response to the Petition for inter partes review, IPR2023-00564, filed by Realtek Semiconductor, Corp. (“Realtek” or “Petitioner”) seeking review of claims 1-9 of United States Patent No. 7,742,053 (the “’053 Patent”). Claims 1 and 5 are independent claims. The Board should deny Realtek’s Petition because it fails to establish the required reasonable likelihood of success under any of the asserted Grounds.

Grounds 1 and 2: Ground 1 relies on the Stuttard reference, and Ground 2 relies on the Stuttard reference as the base reference and the Williams reference as supplementary. Earlier this month, an Initial Determination (“ID”) from the ITC determined that Stuttard does not disclose or render obvious the claim requirement, present in all challenged claims, of “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.” As the ID held, Stuttard processes first only vertex command threads and then only pixel command threads, *i.e.*, it is “phased.” Further, Stuttard uses a SIMD (“single instruction, multiple data”) architecture that can only execute one instruction at a time. Each of these, and in particular both together, counsels against making the modification proposed

by Petitioners here to Stuttard to make its alleged memory device store pixel and vertex command threads simultaneously in two separate portions of memory. Doing so would waste resources for no benefit, as the system cannot operate on pixel commands while in the vertex phase of operation, and vice versa, and so dedicating storage to such inoperable commands is counter to the reference itself.

Although the Board previously found certain challenged claims were rendered obvious in light of a related Stuttard reference, it did so without the benefit of all of the arguments presented here, including new evidence adduced at the ITC hearing, and also because it applied the “broadest reasonable interpretation” claim construction standard, instead of the *Philips* standard that now governs.

Further, as reasoned by the ID, addition of the admitted prior art (“APA”) allegedly disclosing simultaneous storage of both types of command threads does not resolve Stuttard’s deficiencies, because (1) it does not actually disclose the required memory device, as the ID found, and (2) even if such a device were known in the AAPA, using it in Stuttard runs counter to Stuttard for the same reasons as above. Nor does Williams disclose those missing elements, because Williams does not disclose command threads, let alone storing them in separate portions of a memory device.

Even more fundamentally, Petitioner’s own expert testified that Stuttard lacks disclosure of storing a command thread at all, even setting aside whether the

requirements for a plurality of each of vertex and pixel command threads, and for storing them at the same time.

Additionally, Ground 1 fails because Stuttard does not disclose the claimed “arbiter,” which must, in all challenged claims, “select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads.” Stuttard does not disclose nor render obvious storing pixel command threads and vertex command threads at the same time, and so the alleged arbiter cannot select from between them.

Ground 3: Ground 3 fails at least because Petitioner has not done an element-by-element analysis, instead relying on a claim chart without any narrative explanation. That is enough to deny the ground.

Additionally, Ground 3 fails because the combination of Williams and Whittaker, like Stuttard, fails to disclose or render obvious a “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,” for several reasons. Williams does not disclose command threads, nor their separate storage. Additionally, Williams’ fails to disclose the claimed “arbiter,” because the alleged arbiter does not “select” anything, as required by all challenged claims, but merely receives and sorts and/or transfers the data it is handed onto different buses. Ground 3 also fails because Williams fails to disclose the claimed “command

processing engines,” required in all challenged claims. Whittaker is not alleged to fix any of the above deficiencies in Williams, and Whittaker’s purported addition to the combination, the claimed arbiter, fails to disclose the claimed arbiter at least because it does not select between vertex and pixel command threads, as required in all challenged claims.

Accordingly, the Board should deny institution, as all 3 grounds fail.

II. PROCEDURAL OVERVIEW

The ’053 Patent has been subject to two prior validity challenges, both of which ultimately failed. First, on December 10, 2014, LG Electronics, Inc., filed a petition for *inter partes* review, IPR2015-00325, asserting Stuttard ’472 (U.S. Patent No. 7,363,472, which is related to the “Stuttard” reference in the instant proceeding) and Whittaker, along with several references not at issue here. ATI focused its arguments on demonstrating that Stuttard ’472 did not qualify as prior art, which the Board rejected. Ex. 1011 at 53. The Board determined that Stuttard invalidated under the broadest reasonable interpretation (“BRI”) standard. *Id.* at 4, 103. ATI appealed, and the Federal Circuit reversed the Board’s decision as to Stuttard 472’s prior art status and vacated the decision as to invalidity. *ATI Techs. ULC v. Iancu*, 920 F.3d 1362, 1364 (Fed. Cir. 2019).

On May 5, 2022, ATI asserted the ’053 Patent against Realtek and TCL in the ITC, Inv. No. 337-TA-1318, and a parallel action in E.D. Tex., Case No. 2-22-cv-

00134. In the ITC Investigation, Realtek asserted that the '053 Patent was invalid as obvious based on the Stuttard reference at issue here (PCT Pub. No. WO00/62182A2), the combination of Williams and Whittaker, and another reference not at issue here. Less than a week before the evidentiary hearing in the ITC Investigation, Realtek filed the instant petition, asserting obviousness based on Stuttard, Stuttard and Williams, and Williams and Whittaker. Meanwhile, during the ITC Investigation hearing, where only claim 5-6, and 9 of the '053 Patent were still at issue, Realtek asserted invalidity of those claims as obvious based only on Stuttard, and dropped the Williams and Whittaker references.

The Initial Determination of the ITC Investigation, released on July 7, 2023, determined that claims 5, 6, and 9 of the '053 Patent were valid. It held that Stuttard in light of a POSITA's knowledge did not render them obvious. Ex. 2002 (ID) at 180.

III. PRIOR IPR PROCEEDING

There are several reasons there should be a different outcome in this proceeding from the earlier IPR proceeding that found that the related Stuttard '472 reference disclosed the limitations of the challenged claims. The Petition here concedes that Stuttard "does not explicitly disclose [a 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,” but argues for modifications that it states are obvious. Pet. at 20-25.

Since the prior Final Written Decision, the ID, applying the *Philips* standard for claim interpretation, which is applicable here, rather than the BRI standard used in the prior IPR, and applying critical new evidence, including testimony from Petitioner’s expert witness, Dr. Pedram, found that Petitioner’s proposed modification has no motivation and in fact contradicts Stuttard’s operation. Notably, a detailed explanation of Stuttard’s operation was not emphasized in Patent Owner’s prior briefing before the Board, as that briefing was instead focused on the priority date issue. There is now a more fulsome record on those facts, including additional expert testimony from Petitioner’s current expert, who agreed with many of Patent Owner’s positions that the Board previously disagreed with.

The ID found that Stuttard did not invalidate the claims of the ’053 Patent, despite Petitioner strongly emphasizing that the prior Final Written Decision held that the related Stuttard ’472 reference disclosed the elements of the challenged claims. As the ID found, Stuttard does not disclose that vertex and pixel threads are stored in different portions of memory, as Petitioner concedes here. Ex. 2002 (ID) at 81-88. As to obviousness, the ID held that “Stuttard discloses distinct phases of 3D graphics processing (vertex operations, then binning, then pixel operations),” and so it “would not seem to benefit from storing vertex and pixel threads together at the

same time,” *i.e.*, a POSITA would not make the modification Petitioner argues for here. Ex. 2002 (ID) at 85-87.

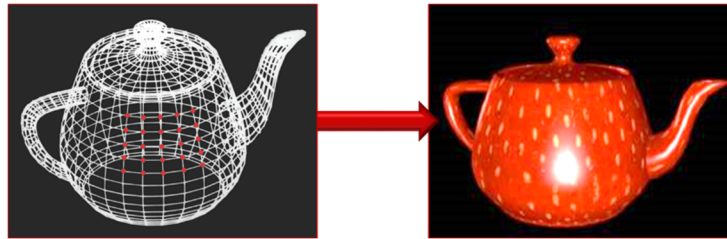
ATI presents additional significant arguments regarding Stuttard that were not raised by prior counsel, which the ID also favorably commented on. For example, Stuttard is a single instruction, multiple data (“SIMD”) system, which means that the entire system can only operate on one instruction at a time. That is fatal to Petitioner’s critical proposed obviousness modification that Stuttard could be modified from its phased system that executes vertex operations first and pixel operations second, in distinct phases—a system that never stores vertex and pixel commands in separate portions of memory and has no reason to—into one that can operate in a non-phased, simultaneous way, storing the two commands in designated separate portions of memory. Fundamentally, that is impossible when the system can only execute one instruction at a time.

Williams and Whittaker do nothing to solve the problems with Stuttard, as explained below.

IV. TECHNICAL OVERVIEW

Graphics processing involves manipulating and rendering 3D graphical objects onto a 2D display screen. Ex. 2004 (Real-Time Rendering) at 11; Ex. 2001, ¶ 32. Specifically, 3D objects (typically represented by a collection of geometrical shapes such as triangles) are converted into a 2D image (typically comprising a

collection of pixels) as shown below. Ex. 2003 (Foley) at 3, 11-15, 17-20; Ex. 2003 (Real-Time Rendering) at 11-13; Ex. 2001, ¶ 32. The 2D image represents a depiction of the 3D objects, rendered from the perspective of a given viewer, and shaded to convey the object's surface material illuminated from a given light source. Ex. 2003 (Foley) at 210-222, 722-741; Ex. 2004 (Real-Time Rendering) at 21-27; Ex. 2001, ¶ 32.



The conversion of a 3D shape consisting of a mesh of triangles into a photorealistic 2D image consisting of an array of pixels happens in a pipeline of stages. Ex. 2003 (Foley) at 806-809, 866-871; Ex. 2003 (Real-Time Rendering) at 12; Ex. 2001, ¶ 33. The first stage is often referred to as “vertex processing” and computes, for each vertex (the point in space corresponding to the junction where neighboring triangles meet) and its 3D position in space, a new 2D position of that vertex in the space of the screen or output image. *See* Ex. 2003 (Foley) at 806-809, 866-871; Ex. 2004 (Real-Time Rendering) at 14-21; Ex. 2001, ¶ 33.

The next stage, “scan conversion,” begins the “rasterization” process by converting the 2D triangle described by the three 2-D vertices output from the vertex

processing stage, into the pixels on the screen that lie within that triangle. Ex. 2003 at 14, 806-809; Ex. 2001, ¶ 34. This scan conversion stage emits all the screen-space pixels that are covered by the triangle. Ex. 2003 at 806-809; Ex. 2004 at 21-22; Ex. 2001, ¶ 34. The final “pixel processing” stage computes the colors of each of these pixels/fragments, and applies textures. Ex. 2003 at 868-872; *see generally* Ex. 2006; Ex. 2004 at 22-23; Ex. 2001, ¶ 34.

V. '053 PATENT OVERVIEW

The '053 Patent is directed to a graphics processing systems, and specifically towards the processing of multiple command threads in parallel. ; Ex. 2001, ¶ 35. The '053 Patent disclosed novel methods for processing multiple threads across one or more hardware processing units with the specialized parallel processing architecture of a graphics processor. *Id.*

For example, in a graphics processing system, the management and control of multiple command threads relating to texture applications is important. Ex. 1001 at 1:22-24; Ex. 2001, ¶ 36. Typically, multiple steps are required to process elements for the application of textures and other processing structures. Ex. 1001 at 1:24-28; Ex. 2001, ¶ 36. A preferred manner for improving the efficiency of graphics processing systems is controlling the flow of the multiple command threads. Ex. 1001 at 1:28-31; Ex. 2001, ¶ 36.

Prior art systems were rigid systems that required providing command threads sequentially. Ex. 1001 at 1:31-48; Ex. 2001, ¶ 37. Another consequence of this architecture was the lack of support for an unlimited number of dependent texture fetches. Ex. 1001 at 2:2-6; Ex. 2001, ¶ 37. Prior to the '053 Patent, conventional systems consisted of two separate command processing engines: a pixel command processing engine that only ran pixel commands and a vertex command processing engine that only ran vertex commands. Ex. 2001, ¶ 37. The vertex command processing engine would first process vertices. *Id.* The triangle primitives defined by these processed vertices would then be scan converted into the pixels that filled them in. *Id.* These pixels would then be processed by the pixel processing engine to determine their final color. *Id.* Since there were 10x to 100x more pixels than vertices, graphics processing could run into a load imbalance causing one of the command processing engines to wait while the other finished its processing. *Id.*

The '053 Patent solves those problems with a novel architecture that uses a single “unified shader processor” (the “command processing engine”) capable of executing both pixel commands and vertex commands:

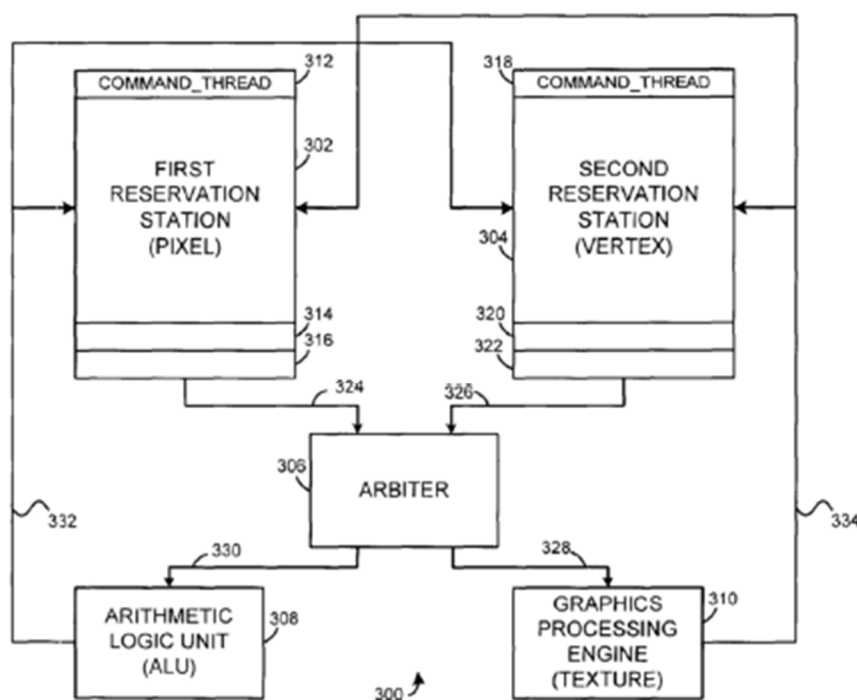


FIG. 4

Id. at Fig. 4; Ex. 2001, ¶ 38. As a result, the arbiter 309 in Fig. 4 above could ensure that the unified shader processor was kept as busy as possible while it completed rendering of the current frame as quickly as possible, *e.g.*, either processing the final pixels of the current frame, or by processing the vertices for triangles in the next frame. Ex. 1001 at 3:16-20; Ex. 2001, ¶ 38.

The “command processing engine” (*i.e.*, the “unified shader processor”) consists of specialized resources, such as an ALU and a texture unit. Ex. 1001 at 2:59-62; Ex. 2001, ¶ 39. The ’053 Patent specification teaches that these resources include parallel processing. Ex. 1001 at 3:40-44; Ex. 2001, ¶ 36. For example, Fig. 3 shown below illustrates an example where the command processing engine

includes multiple ALU's 232 and thread sequencer logic 236 designed to further optimize the parallel execution of command threads. *See, e.g.*, Ex. 1001 at 3:37-47; Ex. 2001, ¶ 36.

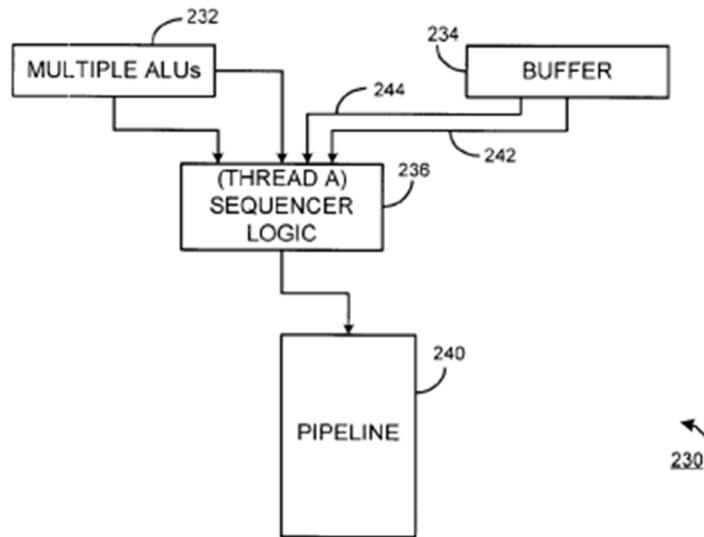


FIG. 3

Id. at Fig. 3.

VI. PRIOR ART OVERVIEW

A. Stuttard (PCT Pub. No. WO00/62182A2)

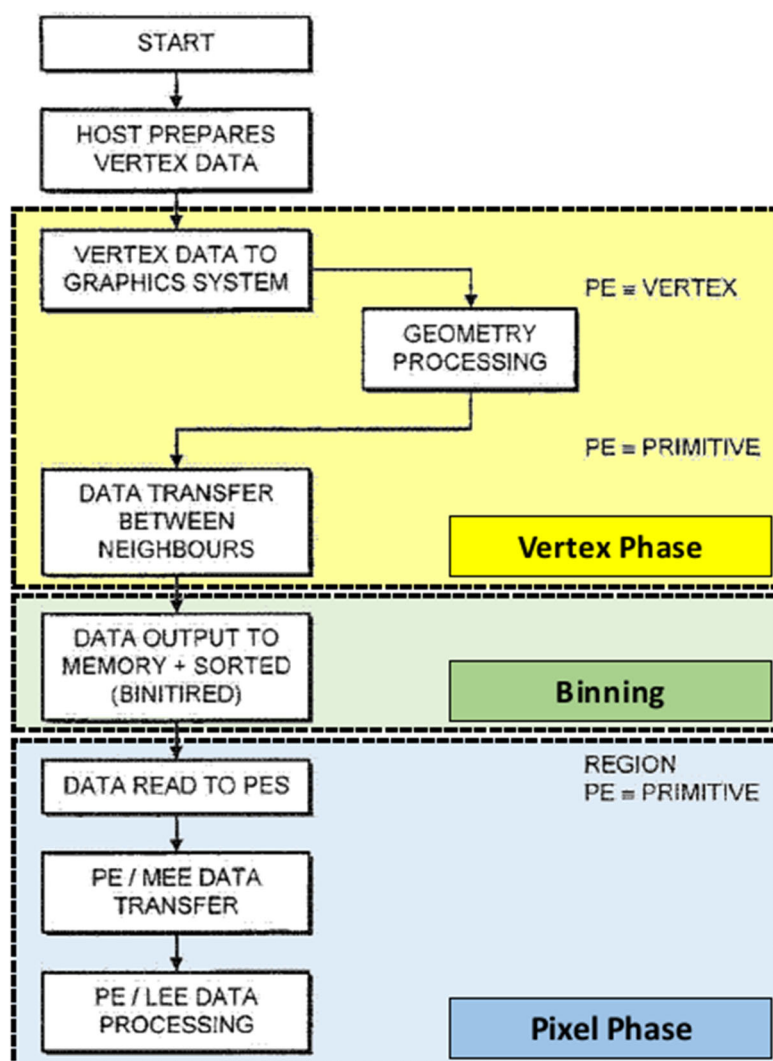
1. Overview

Stuttard (Ex. 1005) is directed to a system which attempts to improve the use of a “single-instruction, multiple data” (“SIMD”) architecture by dividing the processing elements into processing blocks. Ex. 1005 at 2:1-24; Ex. 2001, ¶ 40. Specifically, Stuttard discloses “a SIMD . . . array of processing elements, wherein the processing elements are operably divided into a plurality of processing blocks,

the processing blocks being operable to process respective groups of data items.”
Ex. 1005 at 2:19-23 (emphasis added); Ex. 2001, ¶ 40.

In Stuttard, however, all of the processing blocks of the SIMD operate on “*a common received instruction.*” *Id.* at 2:25-34; *see also id.* at 11:33-35 (“[T]he *same instruction stream* is supplied to all of the processing blocks in the system.”); Ex. 2001, ¶ 41. Thus, at any given time, all processing units within Stuttard are operating on the same instruction. This is distinct from the parallel approach in the ’053 patent, described above. Ex. 2001, ¶ 41.

Also critical to Stuttard’s architecture is that it first operates exclusively on vertices in one phase, and only when that phase is fully complete, then operates exclusively on pixels in a later phase, with no overlap in the execution of these two phases. Ex. 2001, ¶ 42. As explained more fully below, 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.*



Ex. 1005 at Fig. 9 (annotated).

The architecture of Stuttard is illustrated and described in Fig. 1 reproduced below. As shown in Figure 1, Stuttard discloses that the system interfaces with a host system (not shown) via the interface 2. Ex. 2001, ¶ 43. The graphics system 3 includes an embedded processing unit (EPU) 8 “for controlling the overall function of the graphics processor and for interfacing with the host system.” Ex. 1005 at 4:26-

30; Ex. 2001, ¶ 43. The graphics processing capability of Stuttard's system is provided by a processing core 10. Ex. 1005 at 7:25-27; Ex. 2001, ¶ 43.

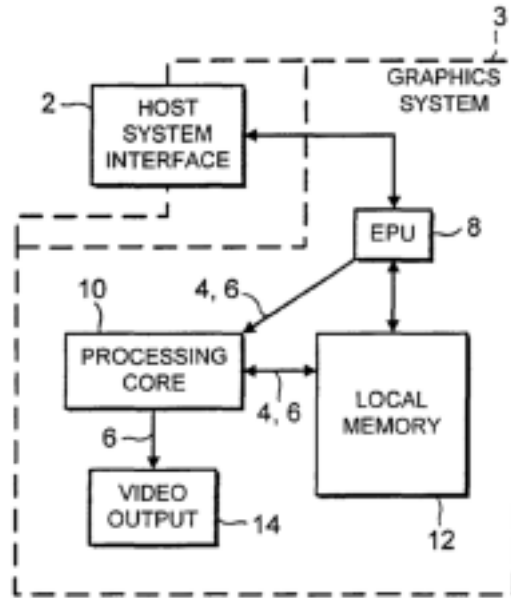


FIG. 1

Ex. 1005 at Fig. 1.

The processing core includes processing blocks 106, which comprise an array 1061 of processor elements (PEs), a mathematical expression evaluator (MEE), and a binning unit. *Id.* at 8:15-9:3; Ex. 2001, ¶ 44. To interface with and manage thread execution by the processing core, Stuttard describes the use of a thread manager 102, an array controller 104, and a channel controller 108, which are illustrated and describe in connection with, for example, Figure 3 (reproduced below). Ex. 2001, ¶ 44.

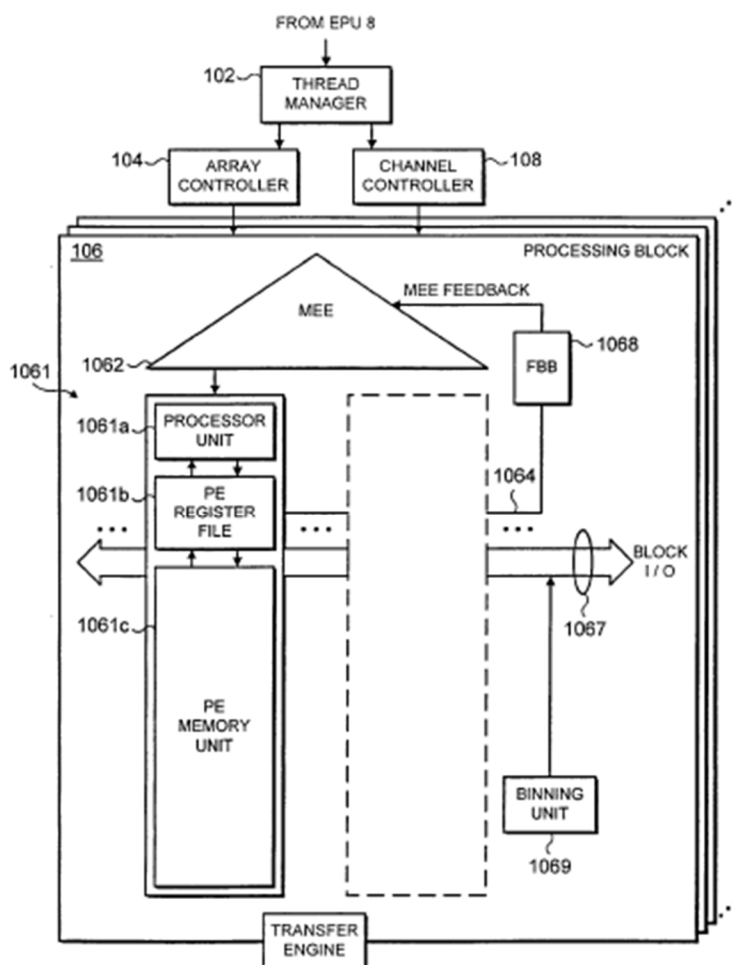


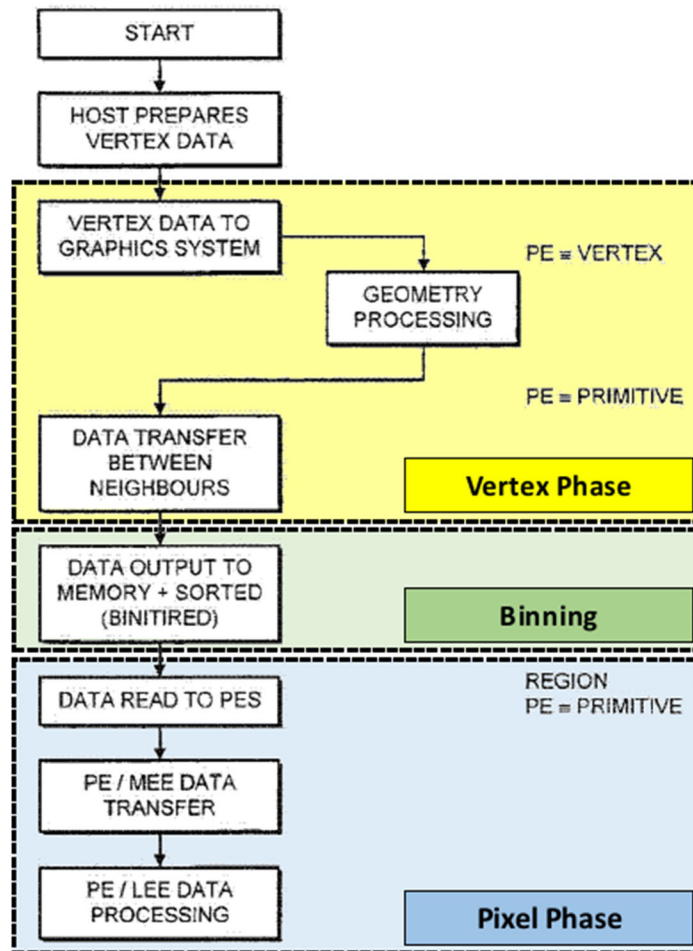
FIG. 3

Ex. 1005 at Fig. 3. “The thread manager 102 is connected to receive control signals from the EPU 8. The control signals inform the thread manager as to when instructions are to be fetched and where the instructions are to be found. The thread manager 102 is connected to provide these instructions to the array controller 104 and to the channel controller 108. The array and channel controllers 104 and 108 are connected to transfer control signals to the processing blocks 106 dependent upon the received instructions.” *Id.* at 8:17-27; Ex. 2001, ¶ 44. Stuttard further discloses

that “[e]ach processing block includes a binning unit 1069” and that “[t]he purpose of the binning unit 1069 is to sort primitive data by region, since the data is generally not provided by the host system in the correct order for region based processing.” Ex. 1005 at 8:34-35; 14:30-35; Ex. 2001, ¶ 44.

2. Phased Architecture

Stuttard is a phased system in which all vertex data and command threads are stored and processed before any pixel data and command threads can be stored and processed:



Id. at Fig. 9 (annotated); Ex. 2001, ¶ 45.

Figure 9 of Stuttard provides steps for processing graphics data using the system. Ex. 1001 at 20:23-25, Fig. 9; Ex. 2001, ¶ 46. The host system prepares data concerning the vertices of the primitive graphical images to be processed and displayed by the system and transfers the data either as a block of vertex data, or vertex by vertex. Ex. 1005 at 20:25-30; Ex. 2001, ¶ 46. The data is loaded into the PEs of the graphics system so that each PE contains data for one vertex, and each PE represents a vertex of a primitive. Ex. 1005 at 20:32-21:1; Ex. 2001, ¶ 46. The received data is processed to transform it from the host system reference space to the required screen space. Ex. 1005 at 21:3-13 Ex. 2001, ¶ 46.

The primitive data is then output from the PEs to local memory so that it can be sorted by region by the binning unit. Ex. 1005 at 21:15-21; Ex. 2001, ¶ 47. The binning unit provides a hardware implemented region sorting system which removes the sorting process from the processing elements, thereby releasing the PEs for data processing. Ex. 1005 at 21:23-26; Ex. 2001, ¶ 47. After the primitive data is sorted, it is read back into the individual PEs, so all the PEs in one processing block contain data concerning respective primitives occurring in a single region. Ex. 1005 at 22:4-10; Ex. 2001, ¶ 47. Next, the PE transfers its data concerning its primitive to the MEE for scan conversion. Ex. 1005 at 22:12-24; Ex. 2001, ¶ 47. Only after scan conversion is the pixel data processed. *See* Ex. 1005 at 22:26-32; Ex. 2001, ¶ 47.

B. Williams (PCT Pub. No. WO00/63770A1)

Williams (Ex. 1006) discloses an apparatus and method for improving a graphics bus. Ex. 2001, ¶ 48. It targets the efficiency of transmitting graphics commands from a CPU to a graphics subsystem, by distributing the transmission across several busses and reconstructing the original order of commands through the use of tags. *See* Ex. 1006 at Abstract; Ex. 2001, ¶ 48. Specifically, Williams identified the speed of graphics data transmitted from a CPU to a graphics subsystem is limited by the bandwidth of and congestion at the peripheral bus. *See* Ex. 1006 at 3:27-4:5; Ex. 2001, ¶ 48. Williams’s disclosure is directed at “increas[ing] the throughput of graphics commands that can be transferred between the system bridge and a graphics subsystem.” Ex. 1006 at 8:4-10; Ex. 2001, ¶ 48.

Williams, in the non-regrouping embodiment, uses a buffer to temporarily store graphics commands. Ex. 1006 at 9:10-12; Ex. 2001, ¶ 49. In this embodiment, the graphics commands can be transferred from the system bridge over bus lines via a sorting unit. Ex. 1006 at 9:8-10; Ex. 2001, ¶ 49. The graphics commands are delivered to the graphics subsystem in the same order as they are transferred from the CPU. Ex. 1006 at 23:14-19 (providing that graphics commands are transferred, “according to the tags, such that the multiple graphics commands regain the specific order (that is, their original order)”; Ex. 2001, ¶ 49.

C. Whittaker (U.S. Patent No. 5,968,167)

Whittaker (Ex. 1007) discloses a generic multi-threaded “data processing management system” to “process multimedia data.” Ex. 1007 at Abstract; Ex. 2001, ¶ 50. This “management system” consists of a “media control core” and “a number of data processing units.” Ex. 1007 at Abstract; Ex. 2001, ¶ 50. The “multimedia core identifies the data processing operation to be executed as well as the resources needed to execute that operation.” Ex. 1007 at Abstract; Ex. 2001, ¶ 50. The media control core “determines which operation has the highest priority” and passes that operation on “to one of the data processing units for execution.” Ex. 1007 at Abstract; Ex. 2001, ¶ 50.

Whittaker was designed to implement an array of independent vector pipeline processors, where each stage of the pipeline performed a fixed function under the control of the data processing core. Ex. 2001, ¶ 50. “[The data processing] units consist of a processing pipeline (data pipeline 10) which will be made up of a number of processing elements such as multipliers, adders, shifters, etc., under the control of an associated data processing core 8 which runs a sequence of instructions to perform a data processing algorithm.” Ex. 1007 at 4:27-33; Ex. 2001, ¶ 50.

VII. CLAIM CONSTRUCTION

For the purposes of this preliminary response, Patent Owner applies Petitioner’s constructions, but does not necessarily agree that they are correct.

Petitioner notes that some terms have been construed by both the Board in the prior IPR and the ALJ in the ITC matter. Pet. at 11. Petitioner states that in those cases the outcome of its Petition is unaffected by which construction is applied. *Id.* The Petition states that “Petitioner will use the narrower of these constructions out of an abundance of caution.” *Id.* However, for at least one term, “command thread,” Petitioner fails to inform the Board of a narrowing construction that it sought that the ALJ adopted that is relevant to the prior art it asserts here.

Further, Patent Owner proposes that the Board adopt constructions of an additional term that the ALJ held were correct in the ID, “at least one memory device . . . operative to store a plurality of pixel command threads and . . . a plurality of vertex command threads,” as explained below.

A. “command threads”

In the ITC litigation, Petitioner argued for a narrowing construction regarding the claim language “at least one memory device . . . operative to store a plurality of pixel command threads and . . . a plurality of vertex command threads,” present in all claims challenged here. Ex. 2002 at 32-39. Petitioner argued that “a complete thread must be stored at a given moment in time” and that it is incorrect that “portions of a complete thread are sufficient so long as the entire thread is stored over a period of time.” *Id.* at 32-33. The ALJ agreed and held that ““at least one memory device . . . operative to store a plurality of pixel command threads and . . .

a plurality of vertex command threads’ means operative to store a plurality of complete sequences of pixel commands and a plurality of complete sequences of vertex commands.” *Id.* at 39.

Patent Owner proposes that the ALJ’s construction, which was sought by Petitioner, be applied here as well.

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

In the Initial Determination, the ALJ held that “it is a correct claim construction that ‘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’ means the memory device(s) are operable to *store both pluralities at the same time.*” Ex. 2002 (ID) at 84. As 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. Ex. 1001 at 2:65-3:3, 3:40-44 (“the sequencer logic 236 receives a first thread 242, and a second thread 244 from the buffer 234, such that the logic 236 may perform simultaneous, interleaved execution of the command threads”), 3:50-53 (“The arbiter selects the command thread by looking at the reservation stations, herein vertex and pixel reservation stations, and picking the first command thread ready to execute.”), 3:62-4:51, 5:10-16, 5:43-56, Fig. 2, Fig. 4, Fig. 5.

As additional rationale, both claim 1 and claim 5 require that the arbiter “select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads.” Ex. 1001 at claim 1, 5; Ex. 2001, ¶ 29. In order for the arbiter to perform the claimed selection, both pluralities of command threads must be stored at the same time, or else both would not be available to be selected from. Ex. 2001, ¶ 29.

Patent Owner proposes that the phrase “at least one memory device . . . operative to store a plurality of pixel command threads and . . . a plurality of vertex command threads” be required to store a plurality of pixel command threads and a plurality of vertex command threads at the same time.

VIII. PETITIONER HAS FAILED TO ESTABLISH THE REQUIRED REASONABLE LIKELIHOOD OF SUCCESS FOR ANY GROUND

A. Ground 1: Stuttard In View of the Knowledge of a POSITA

Petitioner alleges that Stuttard in view of the knowledge of a POSITA renders the challenged claims obvious. However, Stuttard does not disclose the claimed “at least one memory device” because the alleged memory device does not store both pixel and vertex command threads at the same time, a required limitation of all challenged claims. Nor is such a modification obvious in light of Stuttard. Stuttard discloses a *phased* system with a single SIMD, and its alleged memory device would never store pixel command threads during the phase in which it is exclusively

processing vertexes, nor vertex command threads during phase in which it is exclusively processing pixels. Even more fundamentally, Petitioner’s own expert testified that Stuttard lacks disclosure of storing a complete command thread at all, which is required by Petitioner’s construction, even setting aside whether the requirements for a plurality of each of vertex and pixel command threads, and for storing them at the same time. Further, Stuttard does not disclose the claimed “arbiter,” because Petitioner’s alleged arbiter cannot select from the pluralities of both types of command threads, because Stuttard’s memory device does not store both types at the same time.

1. Stuttard 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

Claims 1 and 5 require “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.” Ex. 1001, Claims 1, 5; Ex. 2001, ¶ 53. As the ID found, this claim limitation requires the storage of pixel command threads and vertex command threads in different portions of memory *at the same time*. See Section VII.B.

Petitioner identifies the cache unit 1024 as disclosing the claimed “at least one memory device . . .” of claim 1 and 5. Pet. at 20. However, even Petitioner admits that “*Stuttard’s* FIFO cache unit 1024 does not explicitly disclose ‘a first portion

operative to store a plurality of pixel command threads’ and ‘a second portion operative to store a plurality of vertex command thread.’” Id.; Ex. 2001, ¶ 54.

Petitioner relies on an alleged “obvious” modification to the memory device to store both types of command threads and to do so in separate “portions.” Pet. at 20-25; Ex. 2001, ¶ 54.

a. Evidence Not Before The Prior Panel Confirms Stuttard’s Phased Architecture And SIMD Design, Which Establish There Is No Motivation The Proposed Modification

Petitioner claims that it would be obvious to modify Stuttard to meet the claim limitation, but Petitioner’s arguments fail here, as they did at the ITC. Ex. 2002 (ID) at 81-88; Ex. 2001, ¶ 55. There is no motivation for Petitioner’s proposed modification, for two main reasons. Ex. 2001, ¶ 56. **First**, as explained above in Section VI.A, and as Petitioner’s expert agreed, Stuttard uses a phased system that operates first only on vertex threads and then only on pixel threads, and so there is no reason to store vertex threads and pixel threads at the same time—much less the additional requirement dedicate one “portion” to pixel threads and another “portion” to vertex threads. *Id.* Such resource allocation would waste the valuable, limited resources of a small cache memory to store threads that are not capable of being executed part of the time. *Id.* **Second**, Stuttard’s system is a single SIMD, which only operates on one instruction at a time, as explained above in Section VI.A; Ex.

2001, ¶ 56. And so that additional reason there is no reason or even **ability** to modify Stuttard to be a non-phased system or to dedicate portions of memory to both vertex and pixel threads simultaneously. Ex. 2001, ¶ 56.

The prior Final Written Decision did not consider Stuttard's SIMD configuration, as the evidence was not before it, and also did not have the benefit of the fuller record, including testimony from Petitioner's expert witness, Dr. Pedram regarding the phased nature of Stuttard. Dr. Pedram testified at the ITC hearing:

Q. You will agree with me, at least, that in Stuttard's linear graphics pipeline, the process of pixel processing occurs after geometry processing; correct?

A. I know ***pixel processing occurs after vertex processing***. That, I know.

Ex. 2005 (ITC Hearing Day 4 Tr.) at 994:21-25.

After a multi-day hearing and substantial live expert testimony, the ALJ agreed with Patent Owner's arguments and determined that there is no motivation to modify Stuttard to store both pixel and vertex threads at the same time for the reasons above. Ex. 2002 (ID) at 83-88. The ID noted that "Stuttard discloses ***distinct phases*** of 3D graphics processing (vertex operations, then binning, then pixel operations) and its processing blocks (each containing an array of processing elements ('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 85. The

ID held that Petitioner “and Dr. Pedram do little to dispute this *phased operation*.” Ex. 2002 (ID) at 86. The same is true here.

The ID credited ATI’s expert testimony which provided Stuttard’s phased system would not benefit from storing vertex and pixel threads together at the same time: “[Y]ou would never have pixel threads in there while you were processing vertex threads. . . . And you would never have vertex threads in there while you were processing pixel threads because you would be finishing out the current frame with the pixel threads.” *Id.* at 87. The ID thus concluded that Stuttard’s phased operation “*would not seem to benefit from storing vertex and pixel threads together at the same time*,” rejecting Petitioner’s proposed modification. *Id.* The Board here should hold the same.

The fact that Stuttard’s cache unit 1024 would not be modified to store pixel and vertex command threads at the same time is further bolstered by Stuttard’s SIMD architecture, which the ID also credited. Ex. 2001, ¶ 57. As noted by the ID, Stuttard teaches “[a] data processing apparatus comprises *a SIMD* (single instruction multiple data) array (10) of processing elements,” and further provides that “the same instruction stream is supplied to all of the processing blocks in the system.” Ex. 2002 at 81 (emphasis added), 86. Given Stuttard’s SIMD design, it is incapable of processing both pixel threads and vertex threads at the same time. Ex. 2001, ¶ 57. As Petitioner’s expert admitted, it is well known to a POSITA, that a SIMD

processes only a single instruction at one time. Ex. 2005 at 995:20-22 (providing that SIMD means that the “[s]ame instruction sequence is processing different data items”); Ex. 2001, ¶ 57.

Further, the ID noted that Petitioner’s citations to Stuttard to find motivation for its obviousness argument failed. *See* Ex. 2002 (ID) at 86. In particular, a passage from Stuttard describing the storage of threads relating to 2D activities and 3D activities, which was relied on by Petitioner during the ITC investigation and also relied on in the Petition (Pet. at 20), was unpersuasive because it “does not clearly teach that different kinds of 3D threads, namely vertex and pixel, are stored in different portions of the memory device at the same time, let alone a plurality of each”—just that some type of 2D or 3D threads may be stored. Ex. 2002 (ID) at 85. The ID thus held Petitioner’s motivations to modify Stuttard contradicted Stuttard’s design, and Petitioner repeats those same arguments here, and they should be rejected, now that the Board has a more fulsome record.

b. *Philips* Requires Narrower Interpretation Than The BRI Standard Used In The Prior FWD

Further, the prior FWD applied the then-applicable broadest reasonable interpretation standard for interpreting the claims, as opposed to the current Phillips standard. This difference colors the prior analysis in the FWD, discussed below, regarding the scope of the “memory” term, and also the “arbiter,” which must

“select” from a plurality of vertex command threads and pixel command threads, as explained more fully in Section VIII.A.3.

c. The Analysis In The Prior FWD Was Incorrect

The prior FWD held that it would be obvious to store pixel and vertex commands at the same time in different portions of memory “for ‘faster and convenient storage and retrieval functions, and thereby enhancing the logical layout of the thread processing system,’ taking ‘advantage of Stuttard’s multithreaded architecture that can more efficiently and speedily process graphics data.’” Ex. 1011 at 95-96. But, critically, the Board did not consider that those hypothetical advantages *do not exist* in a phased system, much less one that can only execute one instruction at a time, which is information that was not before it. Instead, those changes waste resources and create overhead for no benefit—a POSITA would recognize that it would waste resources for a phased system to store commands (or set aside storage space for commands and leave it empty) for a phase different from the current phase. For example, storing vertex commands in memory (or setting aside space in memory for vertex commands) while processing pixel commands would reduce the memory available to process pixel commands during the pixel processing phase, which a POSITA would understand could lead to performance degradation not justified by any discernible benefit. Ex. 2001, ¶¶ 58-59. That is, because Stuttard can only operate in vertex or pixel thread mode, it would be an

illogical design choice to modify the memory device to dedicate valuable resources to storing pixel threads when in vertex mode and vertex threads when in pixel mode, because they cannot be used at those respective times. *Id.*

To the extent that the prior FWD’s rationale was that Stuttard is not a phased system, that is incorrect, and even Petitioner’s expert agreed with Patent Owner that it is phased, as explained above. Ex. 1011 at 98; Ex. 2001, ¶ 60. The FWD cited Stuttard’s statement that “[t]he thread scheduler, when running, recalculates which thread should be active,” and concludes that Stuttard is not phased because “[n]othing in Stuttard suggests that, when selecting a pixel thread, the thread manager would disregard all vertex threads, or when selecting a vertex thread, the thread manager would disregard all pixel threads, as alleged by ATI.” Ex. 1011 at 99. The analysis appears to be that the thread manager can choose from pixel and vertex threads, thus switching from one to the other as it chooses. But neither the FWD, LGE, nor Petitioner have pointed to anything indicating that the cache unit stores pixel and vertex command threads at the same time such that the thread manager could choose between them. Ex. 2001, ¶ 60. Indeed, the FWD, LGE, and Petitioner all state that Stuttard would need to be modified for that to occur. *See, e.g.,* Pet. at 51 (arguing it is “obvious”), Ex. 1011 (LGE FWD) at 94 (“LG submits that it would have been obvious to modify Stuttard’s cache unit to store different types of command threads in different portions of memory.”).

To the extent that the prior FWD held that the cache unit could be *modified* to store both type of threads at the same time (as LGE and Petitioner argued), and thus *then* the thread scheduler could choose from both types of threads whenever it wants, thus making the system potentially not phased, that is entirely circular. Ex. 2001, ¶¶ 60-61. That line of logic amounts to (1) reasoning that the memory could be modified simultaneously store both type of command threads for a “faster and convenient storage and retrieval functions” benefit that does not exist in a phased system as explained above, (2) then, due to the newly-modified memory, the system *could* operate in a non-phased way because both command thread types are available in memory to be selected from at a given time, which (3) gives rise to what was originally lacking, a motivation to store both types of command threads at the same time. That is circular and improper.

d. The Alleged AAPA Does Not Change the Analysis

Petitioner’s argument that the alleged AAPA stored both types of threads does not provide a motivation for the proposed modification, as it would still conflict with Stuttard. Pet. at 22; Ex. 2001, ¶ 62. That is, the proposed modification to Stuttard is illogical as explained above, even if it was generally “known” that one could store both types of commands, as Petitioner alleges, because it is a bad design decision given the rest of Stuttard. Ex. 2001, ¶ 62.

Further, even if Stuttard could be combined with the alleged AAPA, which as explained above, there is no motivation to do, as the ID found, Petitioner has failed to prove that the combination of Stuttard and the alleged AAPA discloses “at least one memory device” which stores pixel command threads and vertex command threads *at the same time*. Ex. 2001, ¶ 63. Petitioner asserts that the alleged AAPA discloses storing different types of command threads in different memory portions in a graphics processing system. Pet. at 25. But, as determined by the ID, none of the cited portions of the alleged AAPA disclose pixel and vertex command threads (or any command thread type at all), and most importantly, does not disclose storing them at the same time. *See* Ex. 1001 at 1:42-46; Ex. 2002 (ID) at 87-88 (providing that “the [‘053 patent] passages do not relate to storing vertex and pixel threads together”). In fact, the alleged AAPA states that it has “specifically delineated ALU resource buffers and texture fetch resource buffers such that command threads must be *sequentially* provided between the various buffers.” Ex. 1001 at 1:65-2:1; Ex. 2001, ¶ 63. This sequential provision between different buffers lends further support to the alleged AAPA’s nondisclosure of storing pixel and vertex command threads at the same time.¹ Ex. 2001, ¶ 63.

¹ Patent Owner also disagrees with the inclusion of AAPA, as here it forms the basis of Petitioner’s argument, which is improper under *Qualcomm Inc. v. Apple Inc.*, 24 F.4th 1367 (Fed. Cir. 2022).

2. Stuttard Does Not Disclose or Render Obvious Storing Command Threads

Claims 1 and 5 of the '053 patent require storing “command threads.” As Petitioner previously argued, “a complete thread must be stored at a given moment in time.” *See* Section VII.A. It is insufficient to point to “portions of a complete thread, even if “the entire thread is stored over a period of time.” *Id.*; Ex. 2001, ¶ 64.

But there is no disclosure in Stuttard of storing command threads, much less a plurality of vertex command threads and a plurality of pixel command threads. ; Ex. 2001, ¶ 65. This is separate and apart from the issue in Section VIII.A.1 above regarding whether the two types of command threads are stored at the same time—they are not stored at all. Petitioner points to cache 1024 in Stuttard as the claimed “memory device” of claims 1 and 5, as explained above. Pet. at 20. But cache 1024 only stores individual instructions, *i.e.*, commands, not complete command threads. *See, e.g.*, Ex. 1005 at 11:3-5 (“The thread manager 102 is shown in more detail in Figure 4, and comprises *a cache memory unit 1024 for storing instructions fetched for each thread*”), 11:12-14, 23:3-6, 23:13-16; Ex. 2001, ¶ 65. Petitioner’s expert, Dr. Pedram agreed that an “instruction would be a type of command,” *i.e.* not a command thread. Ex. 2008 at 29:11-14. Therefore, at best, cache 1024 stores instructions that are subparts of a command thread, but as Petitioner argues in its

claim construction, that is insufficient to store “command threads.” *See* Section VII.A; Ex. 2001, ¶ 65. Dr. Pedram is in agreement that Stuttard fails to disclose storing command threads: he testified that Stuttard does not clearly *ever* store a complete command thread in cache 1024 at any given moment in time, which is Petitioner’s construction:

Q So at any given moment in time --

A Yes, sir.

Q -- the cache unit 1024 might never store all of the instructions in a thread?

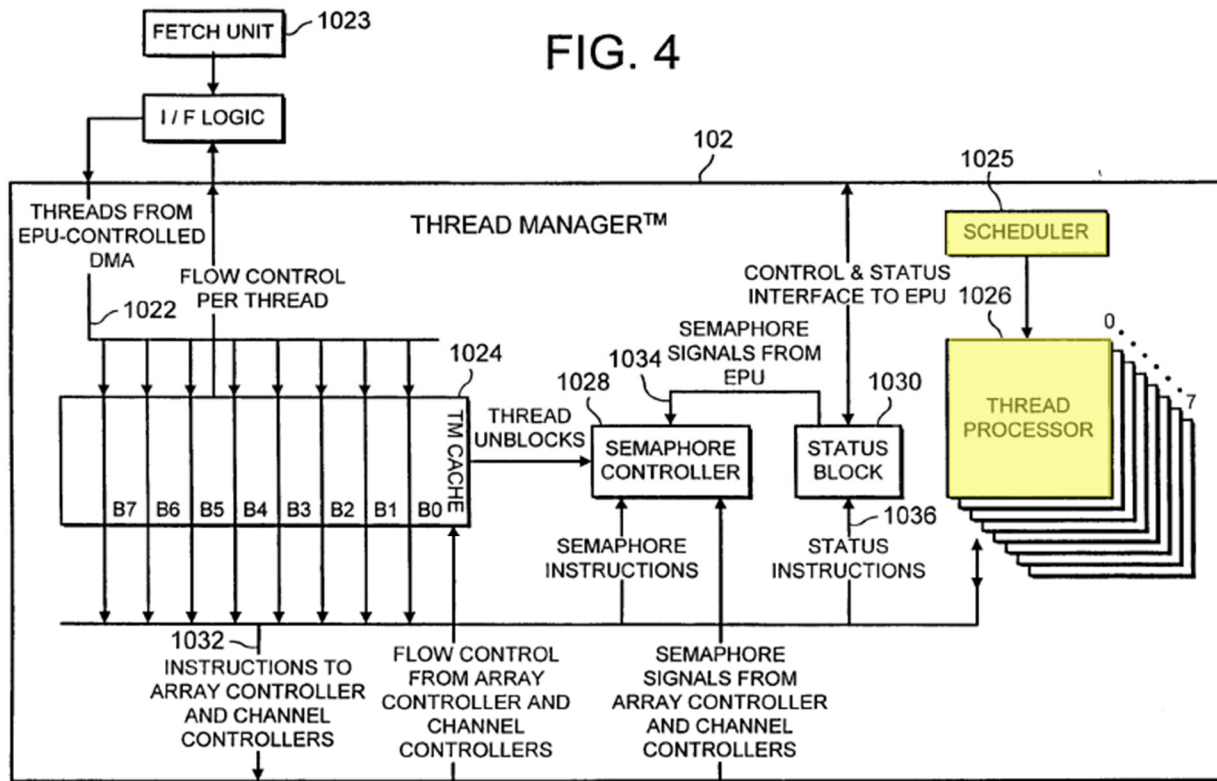
A May or may not.

Ex. 2008 at 183:6-10.

3. Stuttard Does Not Disclose Or Render Obvious An Arbiter That Selects A Command Thread From The Plurality Of Pixel Command Threads Vertex Command Threads

Claims 1 and 5 require “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.” Ex. 1001 at Claims 1, 5; Ex. 2001, ¶ 66. Petitioner argues that “thread scheduler 1025 and thread processors 1026 together correspond to the claimed ‘arbiter’ of claims 1, 2, and 5.” Pet. at 25. Petitioner relies solely on the fact that “[t]he Board has already concluded that the thread manager 102, which includes a thread scheduler 1025 and thread processors

1026 satisfy the arbiter limitations,” in the prior IPR by LG, to conclude that the “arbiter” claim limitation is rendered obvious by Ground 1. *Id.* at 26.



Ex. 1005 at Fig. 4 (annotated).

However, as explained above in Section VIII.A.1 for the “memory device” limitation, there is no point in time when Stuttard’s cache unit 1024 stores both pluralities of pixel command threads and vertex command threads, given Stuttard’s phased operation and SIMD design. *See supra* Section VIII.A.1; Ex. 2001, ¶ 67. This is true even if Patent Owner’s claim construction regarding the memory device is incorrect. Ex. 2001, ¶ 67. Given Stuttard’s express disclosure of a phased system that executes a single instruction at a time, there is no opportunity for the arbiter to

select between at least two pixel command threads and at least two vertex command threads. *Id.*

Notwithstanding the application of a broader claim interpretation standard, the Board’s prior decision was wrong for additional reasons. First, the Board previously held that ATI’s argument ignores that a “thread manager” is used. Ex. 1011 at 98. The Board then leaps from the fact that Stuttard’s cache unit 1024 is capable of storing a plurality of pixel and vertex commands (although it does so at different times) to the conclusion that using the thread scheduler and thread processor enables a non-phased system capable of executing multiple command threads at one time. *Id.* at 99 (“Nothing in Stuttard suggests that, when selecting a pixel thread, the thread manager would disregard all vertex threads, or when selecting a vertex thread, the thread manager would disregard all pixel threads, as alleged by ATI.”).

But that logic is incorrect because when Stuttard’s phased system is processing vertex command threads, only vertex command threads exist in cache unit 1024—pixel command threads do not—and vice versa when the system is processing pixel command threads. Ex. 2001, ¶ 68. Thus, thread manager cannot “select a command thread from either of the plurality of pixel command threads ***and*** the plurality of vertex command threads.” *Id.* Contrary to the FWD’s holding that “[n]othing in Stuttard suggests that, when selecting a pixel thread, the thread manager would disregard all vertex threads,” ***there is no need for a disclosure*** in

Stuttard stating the thread manager “disregards” the other type of thread when in one mode because the other type of thread does not exist in memory.

Viewing the FWD’s reasoning a different way, the Board essentially concluded that because the two types of commands are stored at different times, they could simply instead be stored at the same time, and because there is a thread manager, the reference could be modified from its phased approach to a simultaneous approach as claimed. But that is pure hindsight and circular logic as explained above in Section VIII.A.1.

The PTAB’s reference to the alleged APA (Fig. 1) was incorrect as well, because that structure is fundamentally inconsistent with Stuttard’s phased operation and SIMD architecture that operates on one instruction at a time in a particular phase. Ex. 2001, ¶ 69. Further, the APA is insufficient, because even if the “ALU resource” must be used for vertex commands and the texture fetch resource must be used for pixel commands, the system uses a FIFO – first in first out – for each of the two buffers the arbiter has access to. Ex. 1001 at 1:35-36, Fig. 1; Ex. 2001, ¶ 69. Thus, the arbiter cannot select among the required four items – at least two vertex commands and at least two pixel commands; it only has access to the first item coming out of the buffer and does not select among commands of the same type (*i.e.*, the required “pluralities” component of the claim limitation). Ex. 2001, ¶ 69.

Petitioner has not explained how Stuttard's architecture would be modified to fit the architecture of the APA, or vice versa.

B. Ground 2: Stuttard In View of Williams and the Knowledge of a POSITA

Ground 2 asserts that Stuttard in view of Williams and the knowledge of a POSITA renders the challenged claims obvious. Petitioner attempts to use Williams to solve Stuttard's lack of disclosure of the "at least one memory device" claim limitation. Pet. at 40 ("Specifically, like the admitted prior art in the '053 Patent, Williams discloses and further evidences what was known in the art, *namely, the feature of 'at least one memory device' comprising a 'first portion' and a 'second portion,' each portion respectively storing pixel and vertex command threads.*"). Petitioner relies on two embodiments in Williams, Figure 4 (Pet. at 40-42) and Figure 6 (Pet. at 43), both of which fail.

1. Williams 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

As a preliminary matter, just as with the AAPA, explained above in Section VIII.A.1.d, even if Williams taught the claimed memory device (which it does not, as explained more fully below) proposing to modify Stuttard with Williams fails for the same reasons as with Stuttard alone and Stuttard with the alleged AAPA. *See*

Section VIII.A.1; Ex. 2001, ¶ 71. It proposes a modification to Stuttard that contradicts Stuttard’s fundamental design and has no motivation, as the ID held.

a. Williams Does Not Disclose Or Render Obvious Storing Command Threads

Further, Williams does not disclose the claimed memory device for multiple reasons. Ex. 2001, ¶ 72. Williams does not store “command threads” at all, let alone both pixel and vertex command threads at the same time in separate portions of memory, as required by claims 1 and 5 of the ‘053 patent. *Id.* A “command thread” is a “sequence of commands applicable to the corresponding element,” where the “corresponding element” refers either to vertex data or pixel data. Pet. at 14. That is, there must be some relationship between multiple instructions that are sequenced together, and they must be applicable to particular vertex data or pixel data, to constitute a thread. Ex. 2001, ¶ 71. And as Petitioner previously argued, all challenged claims require storing a complete command thread, not just parts (e.g., individual commands) that over time would constitute a command thread. *See* Section VII.A.

But nothing in Williams describes storing such a sequence, *i.e.*, a command thread as a whole. Ex. 2001, ¶ 71. Williams only stores individual commands, which Petitioner previously stated was insufficient. *Id.* Petitioner identifies the system memory and/or the input buffer of Williams’s Fig. 4 embodiment as

disclosing the claimed “at least one memory device . . . limitation. Pet. at 42; Ex. 2001, ¶ 72.

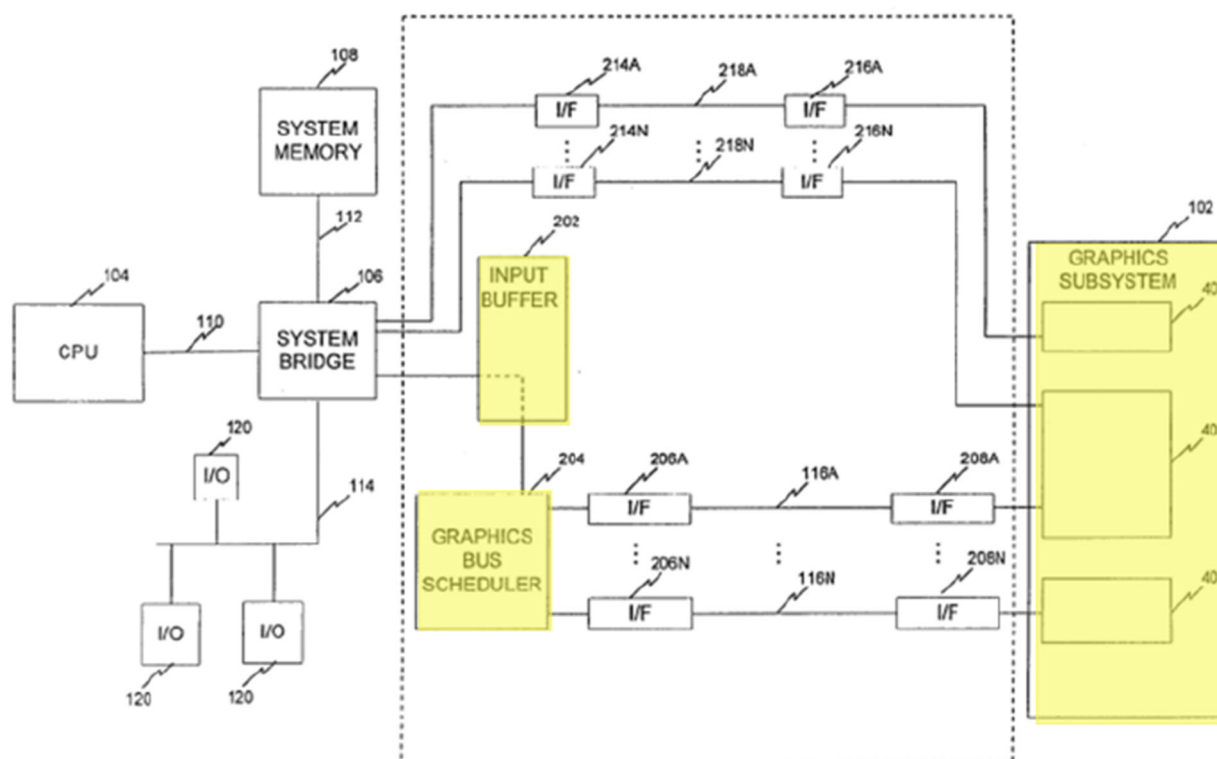


FIG. 4
NON-REGROUPING EMBODIMENT

Ex. 1006 at Fig. 4. Rather, Williams explains that individual “commands” are put into the input buffer 202, without explaining any relationship between multiple commands. Ex. 2001, ¶ 73; Ex. 1006 at 20:14-26. Then “GBS 204 assigns/sorts graphics commands according to their type” and puts them on the appropriate bus. *Id.* Specifically, the “graphics commands are not reordered,” *i.e.*, are not grouped into a “thread.” *Id.*

Williams uses multiple buses to transmit an ordered sequence of graphics commands from the CPU to the GPU. Ex. 2001, ¶¶ 74-76. While multiple buses are used to transmit these commands from CPU to GPU, the order they are processed by the GPU is the same order that they provided from the CPU. *Id.* This demonstrates single-threaded execution, and a POSITA would not have understood such execution to demonstrate multi-threaded execution. *Id.*

The storage devices in the Figure 6 embodiment Petitioner relies on in the alternative, also stores commands and not command threads. Ex. 1006 at 23:1-21.

b. Williams Does Not Disclose Or Render Obvious 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

Moreover, even if Williams were to store command threads, there is no indication of the claimed dedicated “first portion” and a “second portion” of memory, each portion respectively storing pixel and vertex command threads. It simply discloses a buffer for temporary storage of commands. Ex. 1006 at 11:20-24 (“Input buffer 202 . . . temporarily stores graphics commands that are to be transferred from system bridge 106 to graphics subsystem 102.”), 16:28-30, 17:26-27; Ex. 2001, ¶ 77. Petitioner essentially concedes this point and instead argues it would have been obvious to dedicate these first and second portions within memory

“for efficiency of command fetching and transfer, which is Williams’ objective.”
Pet. at 42.

But Petitioner’s alleged motivation for its proposed modification does not actually benefit Williams or Stuttard in any way, and would not be carried out by a POSITA. Petitioner argues that “it was well known that commands stored in consecutive addresses may be *read and written with fewer instructions* than reading commands from (or writing them to) non-consecutive addresses because the latter involves address jumps.” Pet. at 42. But those hypothetical benefits have no relevance to Williams. Ex. 2001, ¶ 78. In Williams, the Graphics Bus Scheduler “sorts the graphics commands according to the type of command” and put them on the appropriate bus line. Ex. 1006 at 20:14-26. That is, the GBS receives the instructions one by one from the input buffer and puts them onto the appropriate bus line. Ex. 1006 at 20:25-26 (“GBS 204 receives multiple graphics commands directly from system bridge 106, or from input buffer 202.”). There are no “address jumps” in Williams, because the GBS does not read from memory addresses to obtain the commands; it simply receives them from buffer as they exit, one after the other. Ex. 2001, ¶ 78. And so the hypothetical benefit of eliminating address jumps and thus using fewer read instructions has no application to Williams, because there are no address jumps or read instructions, and so it does not provide a motivation to modify Williams as Petitioner suggests. Pet. at 42; Ex. 2001, ¶ 78.

In fact, modifying Williams to require reading particular memory addresses in order to retrieve commands would substantially slow the system down. Ex. 2001, ¶ 79. Making the proposed change so that given types of commands were in predetermined memory locations would also render redundant the GBS unit, whose job it is to sort the commands by type based on the randomly-ordered assortment that exit the input buffer. *Id.* At a minimum, Petitioner has failed to meet its burden to show that its proposed system is more “efficient” than the system Williams uses, or any reason a POSITA would make the proposed change. *Id.*

Petitioner’s alternative argument, regarding Figure 6 of Williams, fails for the same reasons. Pet. at 43; Ex. 2001, ¶ 80. Petitioner identifies Williams’ input buffer and output buffer of Fig. 6 (reproduced below) as disclosing the “at least one memory device” claim limitation. Pet. at 43. Petitioner asserts that the input buffer stores pixel commands and the output buffer stores vertex commands, thus meeting the claim requirements. *Id.* This argument suffers from all of the same arguments above, *e.g.*, this disclosure merely stores commands and not command threads. Ex. 2001, ¶ 80. Additionally, there is no indication that they are stored “at the same time,” as some relate to input, before the buses, and some relate to output, after the buses. *Id.*

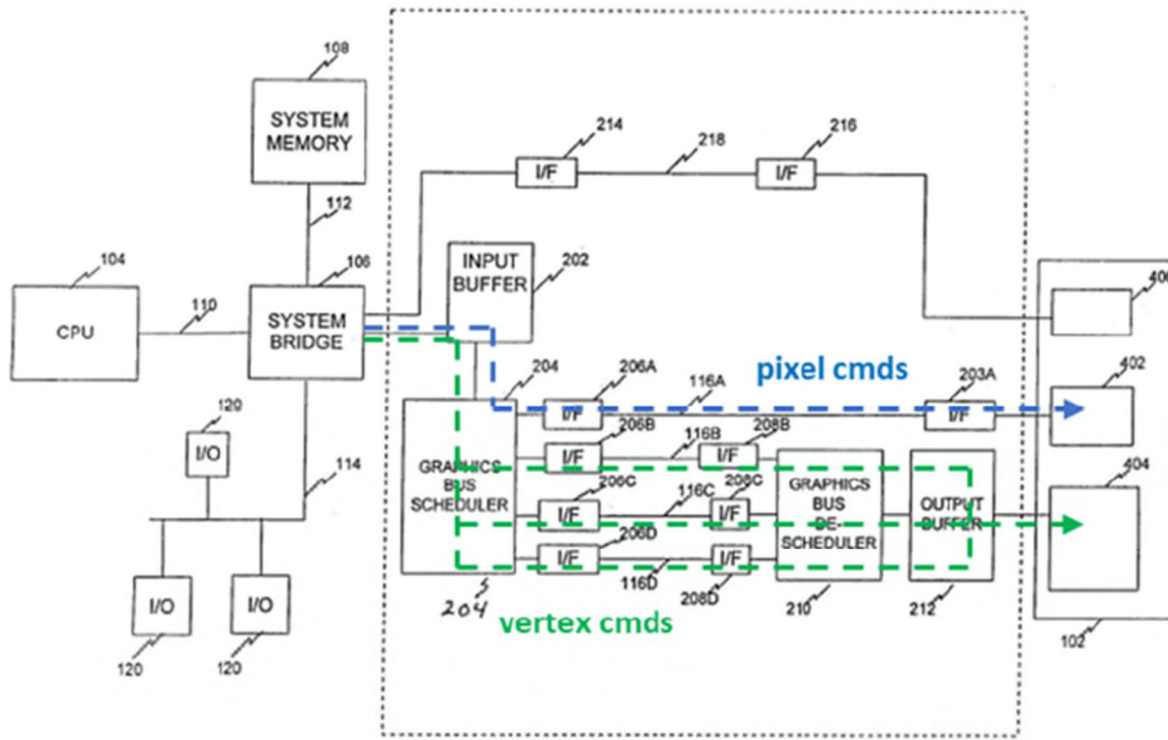


FIG. 6

Ex. 1006, Fig. 6 (annotated by Petitioner); Pet. at 43.

Moreover, these two buffers cannot satisfy the claimed “memory device” as they are not “coupled to” anything that could be an “arbiter,” which is required in the claims 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.” Ex. 1001 at Claim 1; Ex. 2001, ¶ 80. Petitioner’s three-sentence analysis contains no allegations regarding how any alleged “arbiter” would connect to both an input buffer and an output buffer, which serve entirely different purposes, such that it could select between the types of command threads, much less how a system

would be designed such that an arbiter that could select from the above differently-situated input and output buffers would then be coupled to “a plurality of command processing engines . . . each operable to receive and process the command thread.” Ex. 1001 at Claim 5; Ex. 2001, ¶ 80. These two non-analogous buffers simply make no sense as a “memory device” for storing pixel and vertex command threads for execution via an arbiter transmission to command processing engines. Ex. 2001, ¶ 80.

2. There Is No Motivation to Combine Williams with Stuttard

Even if Williams taught or rendered obvious the claimed memory device, which it does not as explained above, a POSITA would not be motivated to combine Williams’s system memory, input buffer, or output buffer with, or as a replacement for, Stuttard’s cache unit 1024, as Petitioner alleges. Ex. 2001, ¶ 81. Petitioner’s motivation to combine argument rests on the two references being both directed to graphics processing, without taking into consideration the nuances of each. *Id.*; see Pet. at 43-44 (proving that both references “share common components such as a host system with a CPU and a graphics subsystem,” etc.).

First, the functionality of Williams’s system memory, input buffer, and output buffer is already redundant of Stuttard’s local memory and system memory devices. Ex. 2001, ¶ 82. A POSITA would not replace Stuttard’s cache unit with William’s system memory where the functionality of these devices is redundant of

functionality already found elsewhere in Stuttard's system. *Id.* For example, Williams' input buffer, which is used "as an interface between system bridge 106 and GBS 204." Ex. 1006 at 18:21-22. This functionality is redundant of the functionality already present in Stuttard's EPU and/or local memory, which also acts as a bridge between the host system and the graphics system and has storage for command threads. Ex. 1005 at 7:1-9, 11:12-14, 23:1-16 ("storing instructions fetched . . . from local external memory 103 or from the EPU 8 by the fetch unit 1023"), Fig. 2; Ex. 2001, ¶ 82. Finally, replacing Stuttard's high speed cache with Williams' lower speed buffer or system memory would render Stuttard inoperable for its intended purpose of enabling high speed instruction execution. *Id.*; *see generally, e.g.*, Ex. 2007.

Moreover, Petitioner's purported motivation to combine speculates that it makes sense to store the different types of commands in different areas of memory, but there is no reason to do that given the role of the GBS within Williams' architecture. Ex. 2001, ¶ 83. Specifically, William provides that upon system memory or input buffer, commands are sorted by GBS onto an appropriate bus. *Id.*; *e.g.*, Ex. 1006 at 11:20-12:2 (providing that GBS 204 "assigns each of the multiple graphics commands" that are "stored within input buffer 202" "to one of the plurality of busses"). If system memory 108 or input buffer 202 already stored different graphics commands in separate portions (which they do not), then the functionality

of GBS 204, which “assigns and/or sorts the graphics commands according to the type of command,” would be redundant because the commands would have already been sorted by type in system memory 108 or input buffer 202. *Id.* at 19:4-5. Nor would there be a need for GBS 204 to assign the graphics commands to different buses based on the type of command. *Id.* Having this kind of redundancy would not achieve Williams’ purported goal of greater efficiency of command fetching and transfer. Ex. 2001, ¶ 83.

For the same reason, Petitioner’s purported hypothetical benefits from grouping similar data at near address spaces in memory also fails. Pet. at 42, 45; Ex. 2001, ¶ 84. Such batching may be useful if a system is capable transferring a batch of data, but that is not what Williams teaches. *See* Ex. 1006 at 11:20-12:2; Ex. 2001, ¶ 84. The graphics commands exit Williams’ input buffer, and get sorted by the GBS on the way out. Ex. 1006 at 11:20-12:2, 18:26-19:11 (“GBS 204 receives multiple graphics commands directly from system bridge 106, or from input buffer 202,” and “assigns each of the multiple graphics commands to one of the plurality of busses 116A-116N.”). That is, each graphics command is individually assigned by GBS 204. Ex. 2001, ¶ 84.

Moreover, even if the system memory and/or input buffer stored a plurality of vertex/pixel command threads in first and second portions, a person of ordinary skill in the art would not be motivated to modify Stuttard’s cache unit 1024 to incorporate

this structure. Ex. 2001, ¶ 85. As previously discussed, Stuttard discloses a phased, SIMD, structure, which is only capable of executing a single instruction at a time. *Id.* As Stuttard's system is incapable of executing pixel threads until it completes all vertex processing, much less at the same time as executing vertex threads, storing both of a plurality of pixel threads and vertex threads in separate first and second portions of the instruction cache 1024 would waste valuable storage resources to store threads that are incapable of being executed. *Id.* As a result of this inefficient use of a caching strategy, the modified version of Stuttard would encounter a greater number of cache misses, making Stuttard's overall system performance worse. *Id.*

C. Ground 3: Williams in View of Whittaker and the Knowledge of a POSITA

Ground 3 asserts that Williams in view of Whittaker and the knowledge of a POSITA renders the challenged claims obvious.

1. Ground 3 Is Procedurally and Substantively Incomplete And Cannot Form A Basis For Institution

As an initial matter, Petitioner fails to provide a complete or sufficient analysis as to why Ground 3 renders the challenged claims invalid. Petitioner states: "Like Ground 2, Ground 3 assumes the analysis for Williams (from Ground 2) and focuses on the additional teachings of Whittaker." Pet. at 51. However, Petitioner did not provide an element-by-element analysis for Williams in Ground 2 (or anywhere in the Petition). In Ground 2, Petitioner relied on Stuttard and only used Williams to

supplement on one element, the “memory device” element and none of the other independent claim limitations. Pet. at 40-50.

In Ground 3, Petitioner similarly does not explain how each independent claim limitation is met by Williams and/or Whittaker. Pet. at 51-61. Instead, Petitioner’s narrative discussion is limited to the phrase “based on relative priorities” within claim 1, not addressing any other limitation of claim 1 or of any other independent or dependent claim. Pet. at 51-53. Petitioner appears to be relying on a chart with bare citations to Williams and Whittaker without any narrative argument or explanation whatsoever. Pet. at 56-61. That is improper and such analysis should be disregarded, and as a result, Ground 3 fails to present a basis on which to institute. 37 C.F.R. § 42.22(a)(2) (“Each petition . . . must include . . . a detailed explanation of the significance of the evidence.”); *Google, Inc. et al. v. EveryMD.com LLC*, IPR2014-00347, Paper 9 (May 22, 2014) at (denying petition where “Petitioners provide a brief summary of Shah's description, and then provide a claim chart that allegedly explains how Shah discloses the claimed subject matter” and “for each limitation of independent claim 1, Petitioners provide quotations and citations of various sections of Shah” because such analysis does not “constitute a detailed explanation of the significance of the quotations, citations, and figures from Shah.”); *GN Resound A/S v. Oticon A/S*, IPR2015-00103, Paper 13 at 6-7 (June 18, 2015) (although quoted passages were present in a claim chart in the petition, the Board

was “not persuaded that bare citations and quotes provide a sufficiently detailed explanation of who the asserted references teach or suggest the claimed limitations”); *Fidelity Nat’l Info. Servs., Inc. v. DataTreasury Corp.*, IPR2014-00489, Paper 9 at 10-11 (Aug. 13, 2014) (denying petition on basis that the citations provided in petition “d[id] not identify sufficiently the portions of the references alleged to teach or suggest the limitations of the challenged claims”).

As to the other reference in this combination, Whittaker, Petitioner only relies on Whittaker to supply claim 1’s requirement that the arbiter select a command thread “based on the relative priorities” of the command thread. Pet. at 51. Whittaker thus is not alleged to cure the deficiencies in Williams as explained above in connection with Ground 2, which includes is the fact that Williams’ memory device does not store command threads nor has an arbiter that selects from both pixel command threads and vertex command threads. Additionally, Williams does not disclose the claimed command processing engines.

2. Williams Does Not Disclose or Render Obvious Storing Command Threads

As explained above in connection with Ground 2, Williams does not disclose storing pixel and vertex command *threads*, as required by all challenged claims. *See* Section VII; Ex. 2001, ¶¶ 71-81, 87. Petitioner presents no new arguments in this Ground to change that analysis and does not rely on Whittaker, the alleged AAPA,

or the knowledge of a POSITA to cure this deficiency. *See generally* Pet. at 51-61.

As such, even for this reason alone, the Board should find that Ground 3 fails.

Additionally, as explained above in Section VIII.B.1, Williams also does not disclose storing any graphic commands in separate portions, at the same time.

3. Williams Does Not Disclose or Render Obvious an Arbiter That Selects a Command Thread

Both independent claims 1 and 5 of the '053 patent require “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.” Ex. 1001 at Claims 1, 5; Ex. 2001, ¶ 88. This requirement being met by Williams is not alleged or analyzed in any narrative portion of the Petition. In the claim chart for Ground 3, without any explanation, Petitioner appears (as nearly as Patent Owner can tell) to argue that GBS 202 is the claimed arbiter. However, Williams’s GBS does not *select* graphics commands (or anything). Rather, as Williams provides, GBS “*receives* multiple graphics commands directly from system bridge 106, or from input buffer 202,” and “assigns each of the multiple graphics commands to one of the plurality of busses 116A-116N.” Ex. 1006 at 18:26-28 (emphasis added); Ex. 2001, ¶ 89. As explained above in Ground 2, the GBS merely receives the commands, one by one, coming out of the input buffer and sorts them; it does not select commands. Ex. 2001, ¶ 89. Moreover, the claim

requires selecting between at least two pixel command threads and two vertex command threads, by virtue of requiring a “plurality” of *each* type, *i.e.*, at least 2 of each type, which is certainly not taught by Williams or Whittaker. *Id.*

4. Williams Does Not Disclose or Render Obvious Command Processing Engines

Claim 5 of the '053 patent requires “a plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread.” Ex. 2001, ¶ 90. As explained above, Williams in view of Whittaker does not disclose pixel command threads or vertex command threads. *Id.* Therefore, Ground 3 does not disclose a command processing engine to receive (and process) the command.

Further, for claim 5, Petitioner cites to the same evidence as claim 2. Pet. at 59. However, claim 5 differs from claim 2 because it requires more than one command processing engine, and also requires each command processing engine is operable to receive and process the command thread. Ex. 1001 at Claims 2, 5; Ex. 2001, ¶ 91. Meanwhile, claim 2 differs from claim 5 because it requires the arbiter is operable to provide the command thread to the command processing engine. Ex. 1001 at Claims 2, 5. Therefore, Petitioner has failed to explain how the plurality of command processing engines receive and process the command thread, which is specific to claim 5.

Petitioner's claim chart as to this element is not explained, but may be intending to identify "functional components" 402 and 404 in Williams as the claimed "command processing engines" because they perform geometry and pixel operations. Pet. at 58-59 (citing Ex. 1006 at 22:3-7). However, these are not "command processing engines" because Petitioner has made no attempt to demonstrate they are coupled to the GBS 204, the alleged arbiter, as they must be per the claims. Ex. 2001, ¶ 92. Nor do these functional components receive both pixel and vertex commands from the GBS 204 (as required by claim 2). *Id.*

Nor are they both operable to process both pixel and vertex commands (as required by claim 5). Ex. 2001, ¶ 93. Williams discloses that functional components 402 and 404 "perform geometry and pixel operations, *respectively*," meaning that each functional component only performs (and thus receives) one operation, but not the other. Ex. 1006 at 22:5-7 (emphasis added); Ex. 2001, ¶ 93. Similarly, Williams also discloses that the graphics commands can be "transferred to the appropriate component/stage of a graphics pipeline 700." Ex. 1006 at 22:7-9; *see also id.* at 25:15-23. Thus, there are only dedicated, type-specific components in Williams alleged to be the claimed command processing engines. Ex. 2001, ¶ 93. Meanwhile, the command processing engines of claim 5 are capable of receiving and processing both the pixel command thread and vertex command thread, depending on which command thread is selected by the arbiter. *Id.* That is, the command processing

engines cannot be type-specific. *Id.* Applying the *Philips* standard, Petitioner's expert, Dr. Pedram agreed that claim 5 excludes type-specific processing engines, that is, command processing engines which processed only pixel or only vertex command threads. Ex. 2005 at 988:8-19. He agreed that each of the command processing engines must be operative to process whichever thread is provided to it by the claimed arbiter, be it a pixel command thread or a vertex command thread. *Id.*

5. Neither Williams Nor Whittaker Discloses or Render Obvious an Arbiter That Selects Command Threads Based On Relative Priorities of the Command Threads

Claim 1 of the '053 patent requires "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." Ex. 1001 at Claim 1; Ex. 2001, ¶ 94. Petitioner admits that Williams does not disclose selection "based on relative priorities" as required by claim 1. Pet. at 51. For this, Petitioner relies on Whittaker's "MCC 2." *Id.* at 51-53. Whittaker, however, also does not disclose pixel command threads or vertex command threads at all, or even different types of command threads, let alone storing in separate portions both pixel and vertex command threads at the same time and then selecting from pluralities of each type of thread. Ex. 2001, ¶ 94. Therefore,

even if MCC 2 selects threads based upon thread priority, nothing in the combination of Williams with Whittaker discloses an arbiter that selects between at least two pixel command threads and at least two vertex command threads. *Id.*

Further, a POSITA would not have been motivated to make the modification to Williams from Whittaker that Petitioner argues for in this element. Ex. 2001, ¶ 95. As explained further below, Williams is directed to overcoming deficiencies of the peripheral bus and increasing the throughput of the graphics data to the graphics subsystem, whereas Whittaker is concerned with the efficient processing of all multimedia data by evaluating resource availability and thread priority. *Id.*

Specifically, the MCC 2 of Whittaker is a fine-grained multithreading processing unit which is in control of all the scheduling and processing of media data. *See* Ex. 1007 at 3:45-49, 4:35-43; Ex. 2001, ¶ 96. In order to select tasks based on task priority, the MCC 2 checks, on every clock cycle, “which of the possible operations it could perform have all the resources available for those tasks to be executed and, of those, which has the highest priority.” Ex. 1007 at 3:56-4:24. The resource checking and priority is illustrated in Figure 6, which explains that for each thread, “global status information is received from the necessary data banks, the necessary address banks, routing control data from the control unit, control status information from control unit 16, and execution dependency data from other processes on which a particular thread is dependent.” *Id.* at 8:17-23, Fig. 6. “All of

this information is sent to a resource checker 81 which combines it with data from 10 ports, the various pipeline data bank status, and the status of various data processing units, . . . for each possible thread.” *Id.* at 8:23-27.

Unlike the MCC 2, Williams’ GBS 204 does not control the components of the graphics subsystem which processes the graphics commands. *See generally* Ex. 1006; Ex. 2001, ¶ 97. Furthermore, in order for GBS 204 to select a thread based the priority of the thread as disclosed in Whittaker, it would also require GBS 204 to perform all the checks and receive all the information disclosed in Whittaker necessary to determine the thread priority. *See* Ex. 1007 at 3:56-4:24, 8:17-35; Ex. 2001, ¶ 97. Petitioner has not adequately explained how a POSITA would understand all the required aspects of MCC 2 to select based on relative priorities would fit into or with GBS 204.

6. There is No Motivation to Combine Williams with Whittaker, Nor Any Reasonable Expectation of Success

A POSITA would not have been motivated to combine Williams and Whittaker. Ex. 2001, ¶ 98. Williams targeted high-performance graphics through bus optimization, seeking to “increase[] the throughput of graphics commands transferred between a system bridge and a graphics subsystem.” Ex. 1006 at Abstract. In contrast, Whittaker sought a cost-saving general multimedia processor “which processes all multimedia data types in a manner that minimises system costs

and provides for future developments in multimedia and the related industry standards,” by prioritizing which threads are being processed. Ex. 1007 at Abstract, 2:4-6; Ex. 2001, ¶ 98. Thus, Williams and Whittaker are aimed at optimizing different components of a graphics processing system, where Williams is directed to the transfer of graphics commands to a GPU, and Whittaker is directed to the prioritization of threads being executed. Ex. 2001, ¶ 98.

Further, a POSITA would not be motivated to combine Whittaker with Williams, because they operate at completely different levels of process analysis. Ex. 2001, ¶ 99. Whittaker’s MCC 2 analyzes threads being processed based on their media needs, *e.g.*, sound, 3-D graphics, 2-D graphics, MPEG video, whereas Williams analyzes only the activity of moving 3-D graphics commands from the CPU to the GPU. *E.g.*, Ex. 1007 at Abstract, 3:36-44, 56-14, 6:42-49, Fig. 1; Ex. 1006 at Abstract. Further, Williams provides assigning graphics commands (*e.g.*, an OpenGL command) to different buffers, and it would not be obvious to a POSITA, even in light of Whittaker, how to assign such priorities to individual OpenGL commands. Ex. 2001, ¶ 99.

A POSITA would not have held a reasonable expectation of success from the combination of Williams and Whittaker. Ex. 2001, ¶ 100. Williams discloses a method for scheduling the transmission of graphics commands across multiple busses by assigning them to different busses based on their type, or order (via first

tagging to indicate the specific order of the commands). *Id.*; Ex. 1006 at 18:14-20, 23:3-21. This approach of Williams conflicts with Whittaker which is designed to be able to ignore the specifics of the media type and instead examines lower-level resource needs, using a single Media Control Core (MCC) that bottlenecks the connection between the data sources and the processing cores. *See, e.g.*, Ex. 1007 at Abstract; Ex. 2001, ¶ 100. This proposed combination would not have provided the throughput sought by Williams, nor the generality sought by Whittaker. Ex. 2001, ¶ 100.

Lastly, Petitioner claims that the Board should find compelling the fact that it previously allowed for a combination of Williams with U.S. Patent No. 7,233,335 (“Moreton”). Pet. at 53. But Petitioner fails to explain how Whittaker is analogous art to Moreton except that both allegedly implement a priority scheme with respect to command threads. *Id.* Without something more, such as analyses as to why the priority schemes of Whittaker and Moreton are analogous, the Board should not take these allegations at face value.

IX. CONCLUSION

For any of the foregoing reasons, the Board should deny institution of the Petition.

Dated: July 27, 2023

Respectfully submitted,

/William A. Meunier/

William A. Meunier (Reg. No. 41,193)
Michael T. Renaud (Reg. No. 44,299)
Adam S. Rizk (Reg. No. 66,867)
MINTZ, LEVIN, COHN, FERRIS,
GLOVSKY AND POPEO, P.C.
One Financial Center
Boston, MA 02111
Telephone: 617-542-6000
Facsimile: 617-542-2241
E-mails: WAMeunier@mintz.com;
ARizk@mintz.com
MTRenaud@mintz.com

Reza Dokhanchy (Reg. No. 62,795)
MINTZ, LEVIN, COHN, FERRIS,
GLOVSKY AND POPEO, P.C.
3580 Carmel Mountain Road, Suite 300
San Diego, CA 92130
Telephone: 858-314-1596
RDokhanchy@mintz.com

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24(d), Patent Owner hereby certifies, in reliance on the word count of the word-processing system (Microsoft Office Word 2010) used to prepare this preliminary response, that the number of words in this paper is 12,666. This word count excludes the tables of contents, tables of authorities, certificate of word count, certificate of service, and table of exhibits.

Dated: July 27, 2023

/William A. Meunier/

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

CERTIFICATE OF SERVICE

I certify that a copy of Patent Owner's Preliminary Response and accompanying ATI Exhibits 2001 through 2008 is being served by electronic mail on the following counsel of record:

Lead Counsel

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

Backup Counsel

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

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

Dated: July 27, 2023

/William A. Meunier/

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