

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
Patent 7,742,053
Title: Multi-Thread Graphics Processing System

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 7,742,053**

TABLE OF CONTENTS

	Page
I. MANDATORY NOTICES	1
A. Real Parties-In-Interest.....	1
B. Related Matters.....	1
C. Lead and Backup Counsel.....	2
D. Service Information.....	2
II. PAYMENT OF FEES	3
III. STANDING	3
IV. REQUEST FOR <i>INTER PARTES</i> REVIEW OF CLAIMS 1-9 OF THE '053 PATENT	3
A. Prior Board Decisions on the '053 Patent.....	4
B. Technology Background	5
C. The Alleged Invention of the '053 Patent.....	7
D. Prosecution of the '053 Patent	9
V. CLAIM CONSTRUCTION	10
A. Construction of Terms Used in the Claims.....	10
B. “arbiter”	11
C. “select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads”.....	12
D. “relative priorities”	13
E. “command processing engine”	13
F. “command threads”	14
G. “memory device”	14
H. “coupled to”	14
VI. SUMMARY OF PRIOR ART.....	15
VII. GROUNDS FOR UNPATENTABILITY	16
A. Ground 1: <i>Stuttard</i> and the knowledge of a person of ordinary skill in the art as evidenced by the Admitted Prior Art of the '053 Patent Render Obvious Claims 1-9.....	16
B. Ground 2: <i>Stuttard</i> and <i>Williams</i> Render Obvious Claims 1-9.....	40

TABLE OF CONTENTS

	Page
C. Ground 3: <i>Williams</i> and <i>Whittaker</i> Render Obvious Claims 1-9.....	51
VIII. CONCLUSION.....	61

TABLE OF AUTHORITIES

Page(s)

Cases

<i>Advanced Micro Devices, Inc. et al. v. TCL Industries Holdings Co., LTD., et al.,</i> Case No. 2:22-cv-00134	1
<i>ATI Techs. ULC v. Iancu,</i> 920 F.3d 1362 (Fed. Cir. 2019)	4, 5
<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same,</i> Inv. No. 337-TA-1318	2, 11
<i>Phillips v. AWH Corp.,</i> 415 F.3d 1303 (Fed. Cir. 2005) (<i>en banc</i>)	10, 11

Statutes

35 U.S.C. § 102(b)	4, 16
35 U.S.C. § 103	1
35 U.S.C. § 103 (a)	16
35 U.S.C. §§ 311-319	1
35 U.S.C. § 315(b)	3

Other Authorities

37 C.F.R. §§ 42.1-.80 & 42.100-.123 <i>et seq.</i>	1
37 C.F.R. § 42.8	1
37 C.F.R. § 42.8(b)(2)	1
37 C.F.R. §42.8(b)(4)	2
37 C.F.R. §42.100(b)	10, 11
37 C.F.R. §42.103	3

37 C.F.R. §42.104(a).....	3
37 C.F.R. §42.104(b)	3

PETITIONER’S EXHIBIT LIST

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

Patent No. 7,742,053
Petition for *Inter Partes* Review

Pursuant to 35 U.S.C. §§ 311-319 and 37 C.F.R. §§ 42.1-.80 & 42.100-.123 *et seq.*, Petitioner respectfully requests *inter partes* review of claims 1-9 of U.S. Patent No. 7,742,053 (“the ’053 Patent”) (“Ex. 1001”), which issued on June 22, 2010. The challenged claims are unpatentable under 35 U.S.C. § 103 over the prior art publications identified and applied in this Petition.

I. MANDATORY NOTICES

Pursuant to 37 C.F.R. § 42.8, Petitioner provides the following mandatory disclosures:

A. Real Parties-In-Interest

Petitioner Realtek is the real party-in-interest.

B. Related Matters

Pursuant to 37 C.F.R. § 42.8(b)(2), Petitioner submits that the ’053 Patent is the subject of a patent infringement lawsuit brought by the Patent Owner, ATI Technologies ULC, against Petitioner in the following cases that may affect or be affected by a decision in this proceeding:

- *Advanced Micro Devices, Inc. et al. v. TCL Industries Holdings Co., LTD., et al.*, Case No. 2:22-cv-00134, now pending in the Eastern District of Texas.

- *Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same*, Inv. No. 337-TA-1318, now pending in the U.S. International Trade Commission.

C. Lead and Backup Counsel

Petitioner designates the following counsel:

LEAD COUNSEL	BACK-UP 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	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

D. Service Information

Pursuant to 37 C.F.R. §42.8(b)(4), Petitioner identifies the following service information: Petitioner consents to service by electronic mail at the following addresses: 3J6PTABDocket@orrick.com; OCHPTABDocket@orrick.com;

Patent No. 7,742,053
Petition for *Inter Partes* Review

sbaik@whitehat.legal; and Realtek-AMD_OHS@orrick.com. Petitioner's Power of Attorney is attached.

II. PAYMENT OF FEES

Pursuant to 37 C.F.R. §42.103, \$23,000 is being paid at the time of filing this petition, charged to Orrick, Herrington, & Sutcliffe LLP: Deposit Account 15-0665. Should any further fees be required by the present Petition, the Patent Trial and Appeal Board ("the Board") is hereby authorized to charge the above referenced Deposit Account.

III. STANDING

Pursuant to 37 C.F.R. §42.104(a), Petitioner certifies that (1) the '053 Patent is available for *inter partes* review; and (2) Petitioner is not barred or estopped from requesting *inter partes* review of claims 1-9 of the '053 Patent on the grounds identified in this Petition. In particular, this Petition is timely filed under 35 U.S.C. § 315(b).

IV. REQUEST FOR *INTER PARTES* REVIEW OF CLAIMS 1-9 OF THE '053 PATENT

Pursuant to 37 C.F.R. §42.104(b), Petitioner requests that the Board finds unpatentable Claims 1-9 of the '053 Patent. Such relief is justified as the alleged invention of the '053 Patent was described by others prior to the effective filing date of the '053 Patent.

A. Prior Board Decisions on the '053 Patent

The Board has already determined that U.S. Patent No. 7,363,472 (“the *Stuttard* Patent”) in combination with the Admitted Prior Art renders obvious claims 1, 2, and 5-7 of the '053 Patent. *See* Ex. 1011. In this Final Written Decision, the Board found the '053 Patent obvious in view of the *Stuttard* Patent as a primary reference, which is the U.S. Patent counterpart to the WO00/62182A2 (“*Stuttard*”) published international application presented in this Petition. *See id.* As expressly acknowledged by the Board, the *Stuttard* Patent is a continuation of and identical to the *Stuttard* foreign application (PCT/GB00/01332) used in this Petition. *See* Ex. 1011 at 13-14. The only difference in the two references is that the *Stuttard* Patent has a filing date of October 9, 2001, and the *Stuttard* foreign application has an earlier filing date of April 7, 2000, making the '182 International publication prior art under § 102(b). *Id.* Thus, the Board’s analysis applies equally to both the *Stuttard* Patent and *Stuttard*’s '182 International publication (*i.e.*, *Stuttard*). Indeed, the Board even referred to both publications in its analysis. *Id.* at 78.

The Federal Circuit overturned the Board’s obviousness determination based on the *Stuttard* Patent, finding that the Patent Owner had shown sufficient evidence that it conceived of the idea of the '053 Patent as of August 24, 2001, and had worked diligently to reduce it to practice prior to the filing date of the *Stuttard* Patent. *ATI Techs. ULC v. Iancu*, 920 F.3d 1362, 1364 (Fed. Cir. 2019). The Federal Circuit

overturned the Final Written Decision based solely on this filing date issue. *Id.* however, as stated above and acknowledge by the Board, the *Stuttard* reference here has a publication date earlier than the August 24, 2001, so *Stuttard* does not have the filing date issues as its U.S. Patent counterpart did in the previous Final Written Decision. *See* Ex. 1011 at 13-14. The '053 Patent is thus obvious under at least *Stuttard* in view of the Admitted Prior Art based on the Board's prior reasoning.

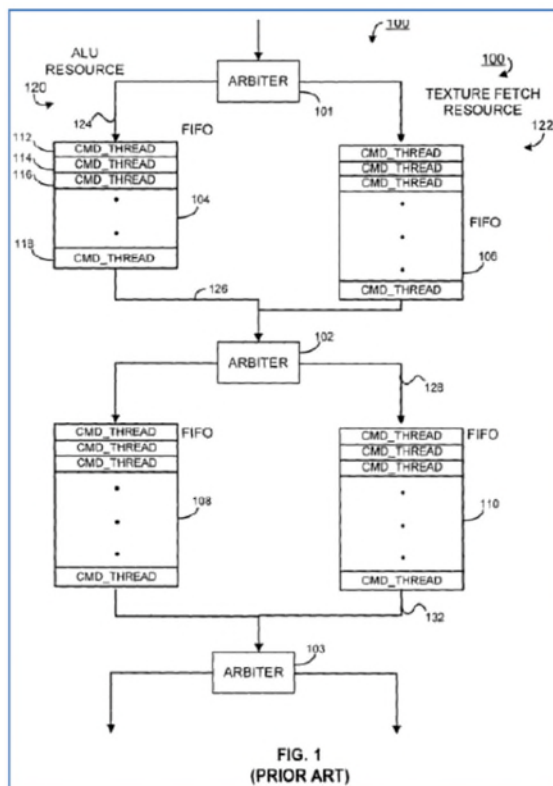
B. Technology Background

In a typical graphics processing system, graphical data such as vertices and pixels are processed through multiple steps providing for the application of textures and other processing instructions. Ex. 1001, 1:24-27. For example, vertex processing occurs when vertices of 3D images are transformed from one coordinate system to a screen coordinate system, and pixel processing occurs when texture mapping and colors are applied to the pixels that have been processed from the vertices. Ex. 1004 (Pedram Decl.), ¶33.

In a typical multithreaded processing system, steps for performing various tasks can be broken down into and controlled by different command threads. Ex. 1004, ¶34. In graphics processing, command threads are sequences of commands used to render graphics and are applicable to specific processing elements. In a typical graphics processing system, pixel command threads are sent to texture fetch

resource devices for the processing of pixel data, and vertex command threads are sent to vertex processing elements for the processing of vertex data.

The different types of command threads are split into separate divisions. Fig. 1 of the '053 Patent illustrates that threads relating to arithmetic and logic operations are grouped into ALU Resource division 120 and threads relating to application of texture are grouped into Texture Fetch Resource division 122.



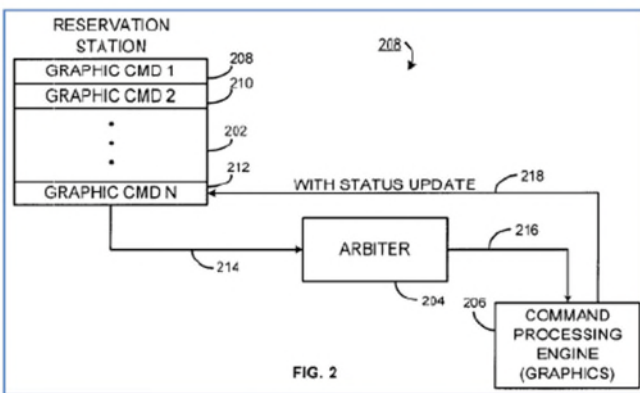
Each of ALU Resource division 120 and Texture Fetch Resource division 122 include buffers (104,106) for storing multiple command threads (*e.g.*, 112, 114, 116, 118). See Ex. 1001, 1:36-42. The command threads are selected for input into the buffers (104, 106) by arbiter 101. See *id.* The command threads stored in the buffers (104, 106) may be withdrawn and provided to specific ALUs or texture fetch processors

for processing. See *id.* at 1:42-48.

In one example, once a command thread is completed, it is sent to a second arbiter 102, which selects and provides “the command thread to either an appropriate texture fetch buffer 110 or an ALU buffer 108.” See *id.* at 1:51-57. Similar steps

are then repeated so that command thread 128 can be sent to another ALU or texture fetch processor for processing and returned to the buffer 108 or 110. *Id.* at 1:58-61. The buffers 108 or 110 could then output command thread 132 “to another arbiter 103 to be provided further along the graphics processing pipeline.” *Id.* at 1:61-64. According to the ’053 Patent, because of the way resource buffers storing the command threads are delineated, threads must be sequentially provided to various buffers. *See id.* at 1:65-2:9. Moreover, dependent fetch instructions that require previous calculations will result in additional delays. *See id.*

C. The Alleged Invention of the ’053 Patent



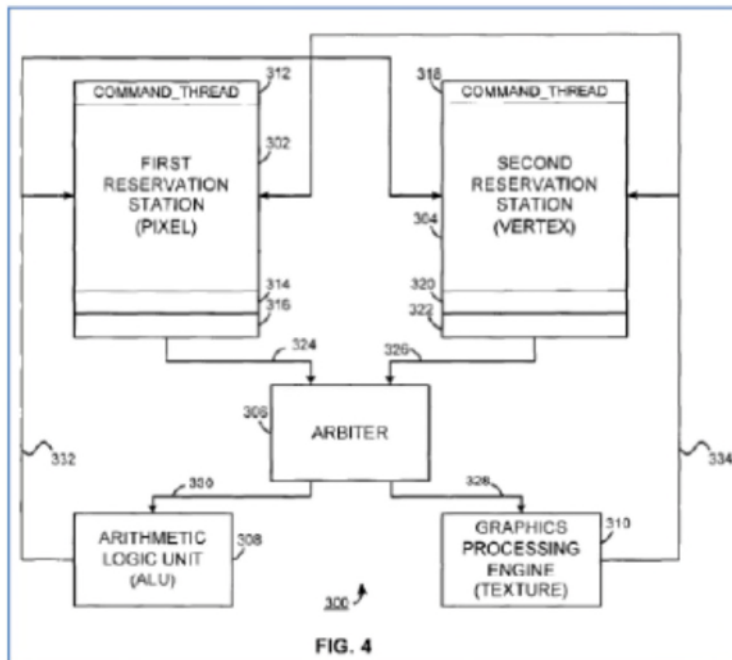
The alleged invention of the ’053 Patent is a graphics processing system including a memory device, an arbiter, and a command processing engine. Ex. 1004, ¶34. An embodiment of the graphics

processing system is illustrated in Fig. 2.

The memory device of the system includes a reservation station for storing a plurality of graphic command threads (208-212). *Id.* at 2:67-3:2. The arbiter 204 is connected to the reservation station to retrieve the command threads from the reservation station and to provide the threads to command processing engine 206.

Id. at 3:08-3:11. Arbiter 204 may retrieve the command threads (208-212) based on a priority scheme. *Id.* at 3:29-30. Once the command processing engine 206 performs the thread command, it provides a status update 218 to the corresponding command thread in the reservation station. *Id.* at 3:12-15; Ex. 1004, ¶35.

Fig. 4 of the '053 Patent discloses another embodiment of the multithread



graphics processing system that includes a plurality of command processing engines:

Similar to the disclosure in the Background of the Invention section, the '053 Patent discloses two reservation stations (302, 304) for storing

command threads for pixel-based command threads and vertex-based command threads, respectively. See Ex. 1001, 3:65-4:4. While not illustrated, an input arbiter is provided to selectively output command threads to each of the two reservation stations (302, 304) based on whether the thread is a pixel or vertex command thread. *Id.* at 4:5-10. From each of the reservation stations, arbiter 306 selects a thread for graphics processing engines 310 and ALU 308. *Id.* at 4:17-21. Upon execution of the associated command of the command thread, the thread is returned to the same

storage location at the reservation stations (302, 304) with its status updated. *Id.* at 4:23-27; Ex. 1004, ¶¶34-35.

D. Prosecution of the '053 Patent

The application which led to the '053 Patent was filed on May 9, 2007, and included original claims 1-19. *See* Ex. 1001, p. 1; Ex. 1002, pp. 16-20. Prior to any substantive examination of the '053 Application, the Examiner issued a Requirement for Restriction/Election of the claims into two groups. Ex. 1002, pp. 56-59. Applicants elected Group II, which included claims 11-19. Ex. 1002, pp. 66-72; Ex. 1004, ¶¶36-37.

On September 9, 2009, the Examiner issued a Non-Final Office Action rejecting all elected claims 11-19 as anticipated or rendered obvious in view of U.S. Patent No. 6,650,327 to *Airey*. Ex. 1002, pp. 76-82. In response to this Office Action, Applicants argued that *Airey* failed to disclose any active select function as required by claims 11-19 and therefore does not disclose or teach all of the features of claims 11-19. *See id.*, pp. 107-110. Applicant did not amend any of the pending claims 11-19. *Id.*, pp. 102-106; Ex. 1004, ¶38.

On February 18, 2010, the Examiner issued a Notice of Allowability indicating claims 11-19 were allowed. Ex. 1002, 117. In the Notice of Allowability, the Examiner indicated the following as Reasons for Allowance:

Reasons for Allowance

3. The following is an examiner's statement of reasons for allowance:

The prior art taken singly or in combination does not teach or suggest, a graphics processing system, among other things, comprising:

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.

The closest prior art, Airey et al. (U.S. Patent No. 6,650,327) fails to teach the above features.

Id. at p. 118. On June 22, 2010, allowed claims 11-19 were issued as final claims 1-9. Ex. 1001, p. 1; Ex. 1004, ¶39.

As discussed above with respect to section IV.A., the Board found claims 1, 2, and 5-7 obvious in IPR2015-00325. The Federal Circuit reversed this decision on the grounds that the asserted prior art was not actually prior art to the '053 Patent. *Id.*; Ex. 1004, ¶40.

V. CLAIM CONSTRUCTION

A. Construction of Terms Used in the Claims

The Board in IPR proceedings applies the same *Phillips* claim construction standard that is used in federal district court cases. *See Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (*en banc*); 37 C.F.R. §42.100(b). A claim subject to IPR

is construed “in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. §42.100(b).

Several terms were previously construed under the broadest reasonable interpretation standard by the Board in IPR2015-00325, and such terms have now also been construed in the related ITC matter under the *Phillips* claim construction standard. Ex. 1010, *Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same*, Inv. No. 337-TA-1318 (Order No. 31, Oct. 18, 2022). Each of the constructions adopted by the Administrative Law Judge (“ALJ”) is broader in scope than the Board’s previous constructions when the Board found claims 1, 2, and 5-7 of the ’053 Patent invalid based on the prior art. See Ex. 1011 at 4-12. Accordingly, adopting either the Board’s original constructions or the ALJ’s broader constructions will have no effect on the outcome. For the purposes of this Petition, Petitioner will use the narrower of these constructions out of an abundance of caution. See Ex. 1004, ¶¶41-42.

B. “arbiter”

The Board previously construed this term to mean “any computer hardware, software, or combination thereof that receives and provides a command thread.” Ex. 1011 at 7. The ALJ in the related ITC matter has also considered the construction of this term, finding that the arbiter only needs to “select a command thread.” Ex.

1010, 17. Because the ITC does not limit this term's construction, the art clearly discloses and/or renders obvious an arbiter in either construction as discussed herein.

Ex. 1004, ¶43.

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

The ALJ in the related ITC matter analyzed this term in his claim construction order and only found that this limitation does not require an arbiter to select a command thread based on state bits associated with each command thread. Ex. 1010, 16. This construction does not add any limitations to the claim language, and this term does not need to be further construed for the purposes of this petition. *See id.* AMD's expert in the ITC matter, argued that “[a]lthough the ALJ does not explicitly construe the term ‘arbiter’ to include this requirement,” this limitation nevertheless means that the arbiter must select from at least four threads, two (or more) pixel threads and two (or more) vertex threads. Such a construction should be rejected because it directly contradicts the plain language of the claims, rendering the phrase “from either of” completely meaningless. Regardless, as the Board previously acknowledged, the Admitted Prior Art describes an arbiter that was known in the art to select a command thread between different types of command threads. Ex. 1011 at 99. Thus, this limitation would have been obvious even under AMD's erroneous construction. Ex. 1004, ¶44.

D. “relative priorities”

The ALJ in the related ITC matter analyzed this term in his claim construction order and only found that the “relative priorities” of the “plurality of pixel command threads and the plurality of vertex command threads” does not refer to whether the command thread is “either a ‘vertex’ priority or a ‘pixel’ priority.” Ex. 1010, 16-17. This construction does not add any limitations to the claim language, and this term does not need to be further construed for the purposes of this petition. *See id.*; Ex. 1004, ¶45.

E. “command processing engine”

The Board previously construed this term “as requiring each command processing engine be coupled to an arbiter and operable to receive and process a command thread selected by the arbiter.” Ex. 1011 at 12. In a claim construction order issued in the related ITC matter, the ALJ rejected the narrowing limitation that the engine must process each command thread selected by and received from the arbiter. Ex. 1010, 17-19. It reasoned that claim 2 does not require the command processing unit to do anything with the command thread upon receiving it from the arbiter, and claim 5 does not require the thread to be received from the arbiter. *Id.* The ALJ did not otherwise come to a final claim construction ruling on this term. *Id.* at 19. Assuming a construction is adopted in the related ITC case, the ALJ

suggests that the construction will be broader than the one previously adopted by the Board. Thus, Petitioner adopts the Board’s previous construction. Ex. 1004, ¶46.

F. “command threads”

The Board previously construed the term “command thread” to “encompass a stream of instructions or a process that is part of a larger process or program.” Ex. 1011 at 6. The ALJ in the related ITC matter construed “command threads” to mean a “sequence of commands applicable to the corresponding element,” with no requirement that the threads comprise state bits. Ex. 1010, 19-20. This construction is sufficiently broad as to not affect Petitioner’s invalidity analysis. Ex. 1004, ¶47.

G. “memory device”

The ALJ in the related ITC matter analyzed this term in his claim construction order, but did not find a reason to construe this term beyond its plain and ordinary meaning. Ex. 1010, 20-21. This construction does not add any limitations to the claim language, and this term does not need to be further construed for the purposes of this petition. *See id.*; Ex. 1004, ¶48.

H. “coupled to”

The ALJ in the related ITC matter analyzed this term in his claim construction order, but only found that the term “coupled to” must be construed broadly and does not require a direct connection. Ex. 1010, 21-22. This construction does not add

any limitations to the claim language, and this term does not need to be further construed for the purposes of this petition. *See id.*; Ex. 1004, ¶49.

VI. SUMMARY OF PRIOR ART

The following documents serve as a basis to show that Petitioner has a reasonable likelihood of prevailing with respect to at least one of claims 1-9 of the '053 Patent. Petitioner provides a detailed explanation of the pertinence and manner of applying the cited prior art to the challenged claims of the '053 Patent in Section VII., *infra*.

Specifically, claims 1-9 of the '053 Patent should be found unpatentable and cancelled in view of the following prior art:

No.	Patent/Publication No.	Filing Date	Date of Issuance	Exhibit No.
1	PCT Pub. No. WO00/62182A2 to Stuttard <i>et al.</i> (Stuttard)	4/7/2000	10/19/2000	1005
2	PCT Pub. No. WO00/63770A1 to Williams <i>et al.</i> (Williams)	4/19/2000	10/26/2000	1006
3	U.S. Patent 5,968,167 to Whittaker <i>et al.</i> (Whittaker)	4/3/1997	10/19/1999	1007

The '053 Patent is a continuation application of application no. 10/673,761, filed September 29, 2003, now U.S. Patent No. 7,239,322. The '053 Patent was filed May 9, 2007, and was issued on June 22, 2010. Further, the Federal Circuit determined that the patentee was able to prove a conception date of August 24, 2001.

Accordingly, *Stuttard* qualifies as prior art under 35 U.S.C. § 102(b) (Ex. 1004, ¶50); *Williams* qualifies as prior art under 35 U.S.C. § 102(b) (Ex. 1004, ¶87); and *Whittaker* qualifies as prior art under 35 U.S.C. § 102(b) (Ex. 1004, ¶121). None of the above references were cited during prosecution.

VII. GROUNDS FOR UNPATENTABILITY

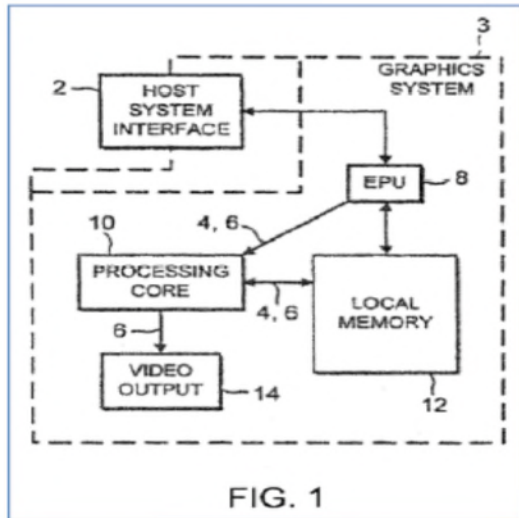
In light of the disclosures detailed below, the '053 Patent is unpatentable for at least the reasons summarized in the chart below and discussed in more detail herein.

Ground	Claims	Proposed Statutory Rejections
1	1-9	Obvious under 35 U.S.C. § 103 (a) over <i>Stuttard</i> in view of the knowledge of a person of ordinary skill in the art as evidenced by the admitted prior art of the '053 Patent
2	1-9	Obvious under 35 U.S.C. § 103 (a) over <i>Stuttard</i> in view of <i>Williams</i>
3	1-9	Obvious under 35 U.S.C. § 103 (a) over <i>Williams</i> in view of <i>Whittaker</i>

A. Ground 1: *Stuttard* and the knowledge of a person of ordinary skill in the art as evidenced by the Admitted Prior Art of the '053 Patent Render Obvious Claims 1-9

Stuttard is directed to a graphics processing system that utilizes multithreading technology to handle input/output (I/O) operations for transferring and processing data. Ex. 1005, 10:12-31; Ex. 1004, ¶51. By keeping processing elements active and overlapping I/O operations with processing operations, *Stuttard*'s system can more efficiently and speedily process graphics data. *Id.* Because this system includes cache memory unit for storing instructions, a thread

scheduler for scheduling the instructions, and a processing block for executing the instructions, it teaches all of the features of claims 1-9. See Ex. 1004, ¶¶51-54.

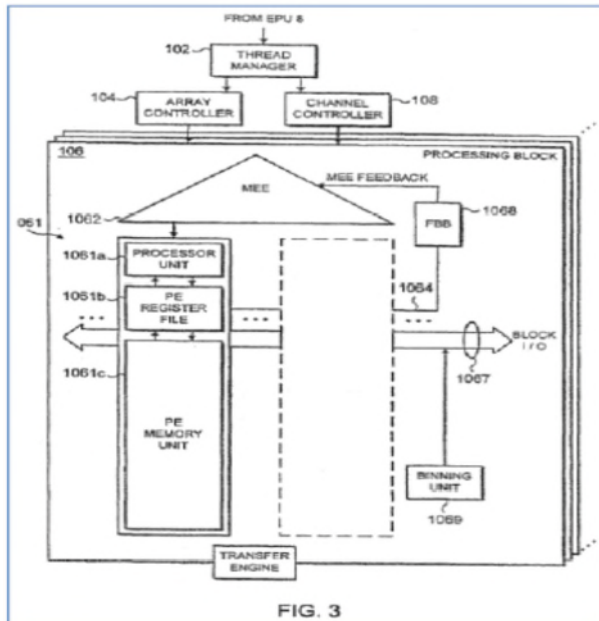


The structure of *Stuttard*'s graphics system is illustrated in Fig. 1.

The graphics system 3 in *Stuttard* includes an embedded processor unit (EPU) 8 that controls how the graphics system functions and interfaces with a host system.

Id. at 4:22-30. The processing core 10 includes a plurality of processing blocks 106 (corresponding to the claimed “*command processing engine*”), which receive vertex data and perform geometry processing by transforming the received vertex data from one coordinate system to another. See *id.* at 8:9-13 and 21:3-5. In addition to processing vertex data, the processing core 10 also carries out operations on pixels. See, *e.g.*, *id.* at 22:4-32; Fig. 9. Simply put, the processing blocks 106 perform the graphics processing. *Id.* at 8:5-7; Ex. 1004, ¶¶51-53.

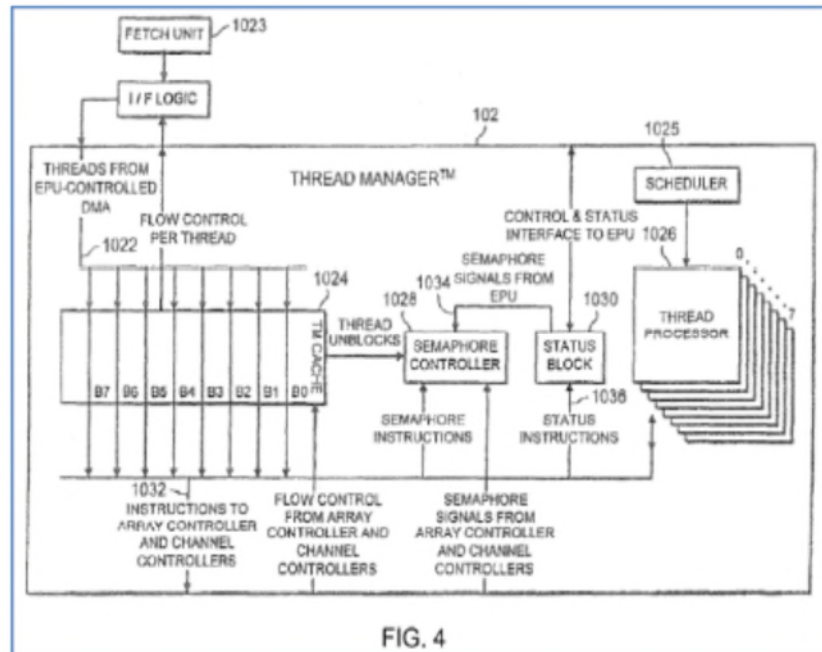
Data and instructions supplied to processing blocks 106 of processing core 10 to control graphics processing operations includes a number of control units such as



thread manager 102, array controller 104, and channel controller 108. *See id.* at 8:5-7; *see also id.* at Fig. 3, reproduced below.

Thread manager 102 provides instructions to both array controller 104 and channel controller 108. For example, instructions from thread manager 102 cause array controller 104

to provide microcode instructions to direct the operation of processing blocks 106. *See id.* at 12:26-28. The components within the thread manager 102 correspond to the claimed “*memory device*” and “*arbiter*” of claim 1 and are further illustrated in Fig. 4:



Thread manager 102 includes a plurality of thread processors 1026 each for a plurality of threads. *Id.* at 11:27-28. The purpose of the thread processor 1026 “is to allow high level flow control to be performed for a thread ... and to assemble instructions to be issued to the array controller 104 and channel controller 108.” *Id.* at 26:3-7. Each thread processor 1026 includes instruction cache 530, which receives addresses for instructions from branch unit 520. *Id.* at Fig. 10(a) and 26:12-14. The thread manager 102 further receives control signals from the EPU 8 that it provides to the array controller 104 and channel controller 108. *Id.* at 8:17-24. Ultimately, instruction assembler 510 within the processor 1026 assembles instructions for channel controller 108 and array controller 104. *Id.* at 27:21-23; Ex. 1004, ¶¶52-53, 60, 63, and 69.

Memory Device

Thread manager 102 includes cache unit 1024 for receiving and storing a plurality of threads with relative priorities that are “assigned to execute 2D (two dimensional) activities, and threads 2 to 7 assigned to execute[] 3D activities (such as calculating vertices, primitives or rastering).” *See id.* at 11:22-25. Cache unit 1024 can be constructed as a series of first-in-first-out (FIFO) buffers, one buffer per thread. *Id.* at 11:3-10. While *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 threads,*” it would have been obvious to a person of ordinary skill in the art that the cache unit 1024 does store both pixel and vertex threads, because *Stuttard*’s processing blocks 106 processes both pixel and vertex data. *See id.* at 20:23-21:7; Ex. 1004, ¶62. Accordingly, it would have also been obvious to one of ordinary skill in the art to modify *Stuttard*’s FIFO cache unit 1024 to separately store pixel and vertex command threads as pixel and vertex command threads are different **types** of data. *See* Ex. 1004, ¶63. It would also have been obvious to modify *Stuttard*’s FIFO cache unit 1024 to separately store and process pixel and vertex command threads for the purpose of faster and convenient storing and retrieving of different types of data, enhancing the logical layout of the thread processing system, and taking advantage of *Stuttard*’s multithreaded system that reduces memory access and

increases operation speed. See Ex. 1005, 12;1-7, 26:3-7; see also Ex. 1004, ¶63. A person of ordinary skill in the art would have also known that pixel and vertex command threads are often stored separately because they correspond to different types of graphics processing tasks. Ex. 1004, ¶63. Further, because this separation was well-known design option at the time, a person of ordinary skill in the art would have expected that modifying *Stuttard* in this way would have been successful. *Id.*, ¶64. Because the pixel and vertex command threads would both be stored in *Stuttard*'s FIFO cache unit 1024, this modification would also satisfy the limitations of claim 4 that require both portions be stored in a single memory device. Further, because a series of FIFOs can replace the cache unit 1024, these FIFOs can be organized into a first and second set of FIFOs that correspond to the separate portions of the memory device. See Ex. 1005 at 23:3-11; Ex. 1004, ¶73. Thus, the different portions in *Stuttard* could be separated into two series of FIFOs that comprise two memory devices as required by claim 3. See *id.* To the extent that these separate memory devices are not explicitly disclosed in *Stuttard*, a person of ordinary skill in the art would have recognized that using one or multiple memory devices would be a simple design choice. See Ex. 1004, ¶73.

The feature of “*at least one memory device*” comprising a “*first portion*” and a “*second portion*” as required by claims 1 and 5 is also found in the admitted prior art in the '053 Patent. The '053 Patent teaches that it was common that memory

devices include first and second portions for storing different types of command threads. For example, the '053 Patent discloses prior-art buffers 104 that store ALU resource command threads, while buffers 106 store texture fetch resource command threads. *See, e.g.*, Ex. 1001, 1:42-48. Similarly, the '053 Patent discloses prior-art memory storage with respect to buffers 108 and 110. *See id.* at 1:49-64. The Board has already agreed that the Admitted Prior Art teaches “stor[ing] different types of graphics command threads in different portions of a memory device.” Ex. 1011 at 95. The Board found that the prior art discloses that “a computer graphics system typically includes basic graphical processing elements, such as vertices and pixels.” *Id.* The Admitted Prior Art further “has one set of buffers for storing ALU resource command threads and another set of buffers for storing texture fetch resource command threads.” *Id.* Because it is well known in graphics processing that arithmetic command threads are commonly vertex data, and pixel data often require fetch or retrieval of texture data, it would have been obvious to one of ordinary skill in the art to modify Fig. 1 of the admitted prior art of the '053 Patent to include only pixel command threads in the ALU resource division (*i.e.*, buffers 104, 108) and only vertex command threads in the texture fetch resource division (*i.e.*, buffers 106, 110) for the purpose of faster access to different types of command threads while also simplifying the storage of threads in memory of a multithreaded processor. *See* Ex. 1004, ¶¶63-66.

Accordingly, it would have been obvious to one of ordinary skill in the art to modify *Stuttard*'s multithreaded system to separately store and process different types of command threads as taught by the background of the '053 Patent for the purpose of utilizing a well-known way of storing different types of command threads and for the purpose of faster access to command threads. It would also have been obvious to modify *Stuttard*'s system to include the thread storage as taught by the background of the '053 Patent for the purpose of taking advantage of *Stuttard*'s multithreaded system that reduces memory access and increases operation speed. See Ex. 1005, 12:1-7; see also Ex. 1004, ¶¶63, 66. One would be motivated to make the modification as both *Stuttard* and the background of the '053 Patent are concerned with the same field of endeavor of processing graphics. See Ex. 1005, 2:17-23 and 4:3-9; Ex. 1001 at 1:24-28. Further, a person of ordinary skill in the art would have recognized this as a routine modification, so it would have expected a reasonable expectation of success. Ex. 1004, ¶64.

Notably, the Board has already agreed that that a person of ordinary skill in the art would have been motivated to modify *Stuttard* with elements known in the art as expressly evidenced by the Admitted Prior Art. Ex. 1011 at 95-97. *Stuttard* teaches a graphics processing system that processes pixel and vertex operations, and the Admitted Prior Art teaches storing different command threads in different memory portions of a graphics processing system. *Id.* at 95. The Board credited Dr.

Bagherzadeh's testimony that a person of ordinary skill in the art would have been motivated to modify *Stuttard*'s cache unit to store different types of threads in different portions of memory, as disclosed in the Admitted Prior Art, for faster and more convenient storage and retrieval functions. *Id.* at 95-96. This modification would take "advantage of *Stuttard*'s multithreaded architecture that can more efficiently and speedily process graphics data." *Id.* The Board further held that Dr. Bagherzadeh's proposed modification was consistent with the Admitted Prior Art's teaching that controlling the flow of multiple command threads would improve the efficiency of the graphics system. *Id.* at 96. As a result, a person of ordinary skill in the art would have been motivated to implement "a memory device having 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 in *Stuttard*'s graphics processing system, in a manner to achieve at [sic] the subject matter of claims 1 and 5." *Id.* at 97.

As discussed earlier, *Stuttard* stores instructions in a FIFO cache unit 1024. Ex. 1005, 11:3-25. Thus, as previously discussed, modifying this FIFO cache unit 1024 to separately store different types of command threads also satisfies the limitations of claim 4 that require first and second portions be stored in a single memory device. Further, as discussed previously, separating the cache unit into a series of FIFOs would yield separate memory devices as would be required by claim

3. Otherwise, as discussed, separating the cache unit of *Stuttard* into two memory devices would have been recognized as a simple design choice. *See* Ex. 1004, ¶73.

In the ITC matter, AMD has argued that the memory device must store both pixel and vertex command threads “at the same time.” Nothing in the claims require any such limitation. AMD’s attempt to import this limitation is an effort to bolster their previously argument to the Board that *Stuttard* should be rejected because it performs operations in “discrete, sequential phases,” *i.e.*, processes only vertex threads followed by pixel threads, but never “at the same time.” The Board previously rejected this argument. Ex. 1011 at 98-99. Regardless, the Admitted Prior Art discloses storing different types of command threads in different memory portions in a graphics processing system, rendering this limitation obvious. Ex. 1001, 1:42-46; Figure 1; Ex. 1011 at 95; Ex. 1004, ¶67.

Arbiter

Thread manager 102 also includes thread scheduler 1025 working in conjunction with thread processors 1026 that schedule and activates the highest priority thread. *See, e.g.*, Ex. 1005, 12:12-18; 23:35-26:32. Based on the thread scheduling, thread processor 1026 “assemble[s] instructions to be issued to the array controller 104 and channel controller 108.” *Id.* at 26:1-5. As such, thread scheduler 1025 and thread processors 1026 together correspond to the claimed “*arbiter*” of claims 1, 2, and 5. The thread manager 102 also utilizes the thread processors 1026

to process the plurality of command threads received. *Id.* at 20:23-21:7. Further, *Stuttard* discloses that the thread manager 102 includes the cache unit 1024 as well, so this “arbiter” is coupled to *Stuttard*’s memory device. *Id.* at 11:3-10. The Board has already concluded that the thread manager 102, which includes a thread scheduler 1025 and thread processors 1026 satisfy the arbiter limitations. Ex. 1011 at 100.

With respect to claim 5, which further requires “*a plurality of command processing engines... each operable to receive and process the command thread,*” *Stuttard* discloses that the thread manager 102 is coupled to multiple processing blocks 106, each with multiple processing element arrays 1061. *See id.* at Fig. 3. *See also id.* at 8:1-27. Each processing element processes both pixel and vertex command threads. *See id.* at 20:32-21:7; 22:26-32. Thus, each processing block 106 comprises a command processing engine. Ex. 1004, ¶¶70-72. The Board has also found that *Stuttard*’s processing blocks disclose a plurality of command processing engines, each operable to receive and process a pixel or vertex command thread. Ex. 1011 at 98. Satisfying claim 5 also satisfies claim 2, as both require the same elements.

In the ITC matter, AMD has argued that each processing engine must be configured to process both pixel and vertex command threads. Again, nothing in the claims require importing any such restriction. Claim 5 requires only that each

command processing engine be operable to receive and process “the” command thread selected by and received from the arbiter. Because the arbiter may select from “either of” pixel or vertex command threads, the claim allows the arbiter to select only pixel command threads or only vertex command threads. Thus, a command processing engine that can process a pixel or vertex command thread, respectively, would satisfy the claim. Alternatively, the “plurality of engines” may include one or more engines that process only pixel command threads when selected, with another one or more engine that processes only vertex command threads when selected. Regardless, as discussed further herein, *Stuttard* discloses both ALU and texture processing engines, which are capable of processing both vertex and pixel operations. Ex. 1004 ¶79; Ex. 1011 at 101.

Arithmetic Logic Unit

With respect to claim 6, *Stuttard* discloses that the processor unit 1061a, which forms part of the processing element 1061, as having “an arithmetic logic unit ... which is connected to receive data values.” *Id.* at 47:24-28; *see also id.* at Fig. 11.

Texture Processing Engine

With respect to claim 7, *Stuttard* discloses that “lighting and shading ... [are] performed to produce data depending upon the chosen viewpoint.” *Id.* at 21:3-7. As such, one of ordinary skill in the art would recognize that any lighting and shading performed necessarily requires some type of texture processing engine. *See* Ex.

1004, ¶¶83-84. The Board has already found that *Stuttard* teaches a texture processing engine based on the knowledge of a person of ordinary skill in the art. Ex. 1011 at 100-102. The Board recognized that “[t]here is no dispute that the use of a texture processing engine in a graphics processing system was known in the art at the time of the invention.” *Id.* at 101; Ex. 1001, 1:32-48. *Stuttard*’s system processes both vertex and pixel operations, and it performs “three dimensional geometry, view, lighting and shading.” Ex. 1011 at 101; Ex. 1005, 20:23-22:32. Vertex commands can also involve texture operations, and the majority of pixel command threads involve texture operations. Ex. 1011 at 101. The Board found that, therefore, a person of ordinary skill in the art “would have recognized that *Stuttard*’s graphics processing system includes a texture processing engine to process pixel and vertex command threads.” *Id.*

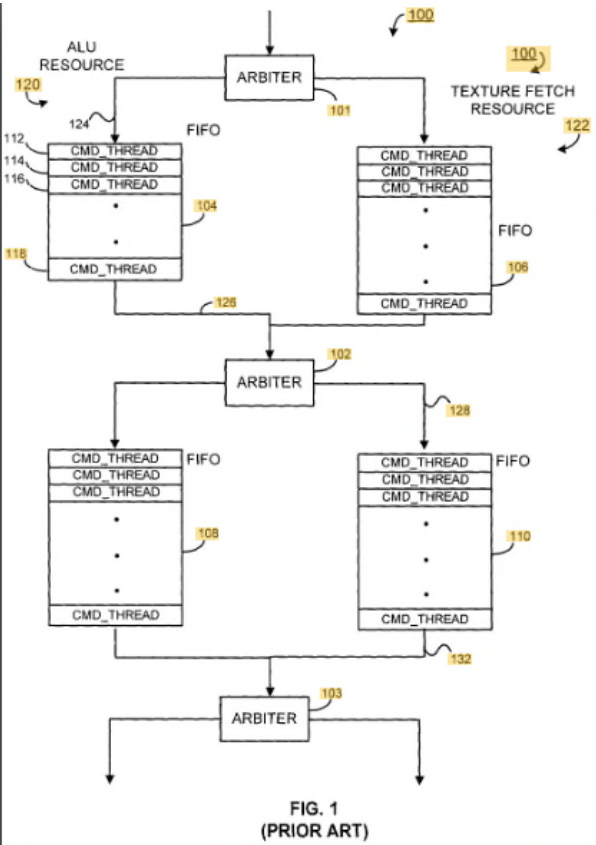
Accordingly, and as explained by the claim chart below, *Stuttard* in view of the AAPA teaches all limitations of claims 1-9 of the ’053 Patent. *See* Ex. 1004, ¶¶50-86.

Claim	<i>Stuttard</i> (Ex. 1005)
1[pre]. A graphics processing system comprising:	“Figure 1 is a system level block diagram illustrating a graphics data processing system 3. The system 3 interfaces with a host system (not shown), such as a personal computer or workstation, via an interface 2. Such a system can be provided with an embedded processor unit (EPU) for control purposes. For example, the specific graphics system 3 includes an embedded

Claim	<i>Stuttard</i> (Ex. 1005)
	processing unit (EPU) 8 for controlling the overall function of the graphics processor and for interfacing with the host system. The system includes a processing core 10 which processes the graphical data for output to the display screen via a video output interface 14. Local memory 12 is provided for the graphics system 3.” Ex. 1005, 4:21-32. <i>See</i> Ex. 1004, ¶¶55-58.
[1.a] 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; and	<u><i>Stuttard</i></u> “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. The cache unit 1024 could be replaced by a series of first-in-first-out (FIFO) buffers, one per thread. The thread manager also includes an instruction fetch unit 1023, a thread scheduler 1025, thread processors 1026, a semaphore controller 1028 and a status block 1030. Instructions for a thread are fetched from local memory or the EPU 8 by the fetch unit 1023, and supplied to the cache memory 1024 via connecting logic.” Ex. 1005, 11:3-14. “The threads are assigned priorities relative to one another. Of course, although the example described here has eight threads, any number of threads can be controlled in this manner. At any particular moment in time, each thread may be assigned to any one of a number of tasks. For example, thread zero may be assigned for general system control, thread 1 assigned to execute 2D (two dimensional) activities, and threads 2 to 7 assigned to executing 3D activities (such as calculating vertices, primitives or rastering)” <i>Id.</i> at 11:16-25 “Figure 9 illustrates simplified steps in a graphics data processing method using the system of Figures. 1 to 8. The host system prepares data

Claim	Stuttard (Ex. 1005)
	concerning the vertices of the primitive graphical images to be processed and displayed by the graphics system. The data is then transferred, either as a block of vertex data, or vertex by vertex as it is prepared by the host system to the graphics system. The data is loaded into the PEs of the graphics system so that each PE contains data for one vertex. Each PE then represents a vertex of a primitive that can be at an end of a line or part of a two dimensional shape such as a triangle. The received data is then processed to transform it from the host system reference space to the required screen space. For example, three dimensional geometry, view, lighting and shading etc. is performed to produce data depending upon the chosen viewpoint.” <i>Id.</i> at 20:23-21:7. “Once each pixel is determined to be outside or inside the triangle (primitive) concerned, the processing for the primitive can be carried out only on those pixels occurring inside the primitive. The remainder of the PEs in the processing block do not take any further part in the processing until that primitive is processed.” <i>Id.</i> at 22:26-32. “A detailed description will now be given of the thread manager 102, which as mentioned above with reference to Figure 4, comprises a cache memory unit 1024 for storing instructions fetched for each thread. The cache unit 1024 could be replaced by a series of first-in-first-out (FIFO) buffers, one per thread. The thread manager also includes an instruction fetch unit 1023, a thread scheduler 1025, thread processors 1026, a semaphore controller 1028 and a status block 1030.” <i>Id.</i> at 23:3-11.

Claim	<i>Stuttard</i> (Ex. 1005)
	“The purpose of the thread processor 1026 is to allow high level flow control to be performed for a thread, (such as looping and conditional branches), and to assemble instructions to be issued to the array controller 104 and channel controller 108.” <i>Id.</i> at 26:3-7. <i>See</i> Ex. 1004, ¶¶59-65. <u>Admitted Prior Art</u> “In a typical graphics processing system, the processing elements, such as vertices and/or pixels, are processed through multiple steps providing for the application of textures and other processing instructions, such as done through one or more arithmetic logic units (ALU).” Ex. 1001, 1:24-28. “The command thread 118 may then be withdrawn from the reservation stations 104 and 108 for the purpose of being provided to an ALU (not shown) and the command threads within the texture fetch resource division 122 maybe withdrawn from the reservation stations 106 and 110 to be provided to a texture fetch processors (not shown). In the prior art embodiments of FIG. 1, the first buffer 104 receives an input command 124 and outputs a completed command thread 126 to the second arbiter 102. In one embodiment, the command thread may include an indicator, such as a flag, indicating when the access to the ALU resources has been completed for the associated command. The arbiter 102 receives the input command 124 and thereupon provides, in due course, the command thread to either an appropriate texture fetch buffer 110 or an ALU buffer 108. Thereupon, the steps are repeated where an output thread command 128 is provided to another ALU (not shown) or texture fetch

Claim	Stuttard (Ex. 1005)
	processor (not shown) and returned to the buffer 108 or 110. The buffer 110 also produces the output 132 which is a command thread. The output 132 may be provided to another arbiter 103 to be provided further along the graphics processing pipeline.” <i>Id.</i>, 1:42-64.  FIG. 1 (PRIOR ART)
[1.b] 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	“Within the thread manager 102, the status block 1030 receives status information 1036 from each of the threads. The status information is transferred to the thread scheduler 1025 by the status block 1030. The status information is used by the thread scheduler 1025 to determine which thread should be active at any one time. Core instructions 1032 issued by the thread manager 102 are sent to the array controller 104 and the

Claim	<i>Stuttard</i> (Ex. 1005)
the plurality of vertex command threads.	channel controller 108 (figure 3).” Ex. 1005, 12:13-22. “At a given time, only one thread is executing, and the scheduling of the time multiplexing between threads is determined by the dynamic conditions of the program execution. This scheduling is performed by a thread scheduler in the thread manager 102, which ensures that each processor block 106 is kept busy as much as possible. The switching from one thread to another involves a state saving and restoring overhead. Therefore, the priority of threads is used to reduce the number of thread switches, thereby reducing the associated overheads. Core instructions issued by the thread manager 102 are sent to one of two controller units, the array controller 104 or channel controller 108.” <i>Id.</i> at 23:18-32. “Determining which Thread should be Active The thread scheduler, when running, recalculates which thread should be active whenever one of the following scheduling triggers occur: A thread with higher priority than the current active thread is READY, or The thread is (not Ready) and YIELDING... The instruction assembler 510 assembles instructions for the various controllers such as channel controller 108 and array controller 104. Most instructions are not modified and are simply passed on to the respective controllers. However, sometimes fields in the various instructions can be replaced with the contents of the general registers. The instruction assembler 510 does this before passing the instruction to the relevant controller. The instruction assembler 510 also calculates the yield status, the wait status and the controller signal status sent to the

Claim	Stuttard (Ex. 1005)
	enable unit 500 and the scheduler in the thread manager 102.” <i>Id.</i> at 23:34-27:32. “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. The cache unit 1024 could be replaced by a series of first-in-first-out (FIFO) buffers, one per thread. The thread manager also includes an instruction fetch unit 1023, a thread scheduler 1025, thread processors 1026, a semaphore controller 1028 and a status block 1030.” <i>Id.</i> at 11:3-10. “The threads are assigned priorities relative to one another. Of course, although the example described here has eight threads, any number of threads can be controlled in this manner. At any particular moment in time, each thread may be assigned to any one of a number of tasks. For example, thread zero may be assigned for general system control, thread 1 assigned to execute 2D (two dimensional) activities, and threads 2 to 7 assigned to executing 3D activities (such as calculating vertices, primitives or rastering)[.] In the example shown in Figure 4, the thread manager includes one thread processor 1026 for each thread. 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 the processing blocks 106 can be active for as much time as possible. In this particular example the same instruction stream is supplied to all of the processing blocks in the system.” <i>Id.</i> at 11:16-35. “It will be appreciated that the number of threads could exceed the number of thread processors, so that each thread processor handles control of

Claim	<i>Stuttard</i> (Ex. 1005)
	more than one thread. However, providing a thread processor for each thread reduces the need for context switching when changing the active thread, thereby reducing memory accesses and hence increasing the speed of operation.” <i>Id.</i> at 12:1-12:7. “Figure 9 illustrates simplified steps in a graphics data processing method using the system of Figures 1 to 8. The host system prepares data concerning the vertices of the primitive graphical images to be processed and displayed by the graphics system. The data is then transferred, either as a block of vertex data, or vertex by vertex as it is prepared by the host system to the graphics system. The data is loaded into the PEs of the graphics system so that each PE contains data for one vertex. Each PE then represents a vertex of a primitive that can be at an end of a line or part of a two dimensional shape such as a triangle. The received data is then processed to transform it from the host system reference space to the required screen space. For example, three dimensional geometry, view, lighting and shading etc. is performed to produce data depending upon the chosen viewpoint.” <i>Id.</i> at 20:23-21:7.” <i>See also, e.g., id.</i> at Fig. 4; Ex. 1004, ¶¶68-69.
2. The graphics processing system of claim 1, further comprising: a command processing engine, coupled to the arbiter, wherein the arbiter is further operable to provide the	“Figure 3 shows the processing core in more detail. 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

Claim	<i>Stuttard</i> (Ex. 1005)
command thread to the command processing engine.	104 and 108 are connected to transfer control signals to the processing blocks 106 dependent upon the received instructions.” Ex. 1005, 8:17-27. “Each processing element (PE) 1061 includes a processor unit 1061a for carrying out the instructions received from the array controller, a PE memory unit 1061c for storing data for use by the processor unit 1061a, and a PE register file 1061b through which data is transferred between the processor unit 1061a and the PE memory unit 1061c. The PE register file 1061b is also used by the processor unit 1061a for temporarily storing data that is being processed by the processor unit 1061a.” <i>Id.</i> at 9:11-20. “As will be explained in greater detail below, the processing core 10 includes a number of control units: thread manager 102, array controller 104, channel controller 108, a binning unit 1069 per block and a microcode store 105. These control units control the operation of a number of processing blocks 106 which perform the graphics processing itself.” <i>Id.</i> at 8:1-7. “In the example shown in Figure 2, the processing core 10 is provided with eight processing blocks 106. It will be readily appreciated that any number of processing blocks can be provided in a graphics system using this architecture.” <i>Id.</i> at 8:9-13. “In the present system the processing blocks 106 are provided with opcodes (instructions) and operands (data items) for the expression evaluator separately from one another. Previously, instructions and data are provided in a single instruction stream. This stream must be produced during processing which can result in a slowing of processing speed, particularly when

Claim	<i>Stuttard</i> (Ex. 1005)
	the operands are produced in the array itself.” <i>Id.</i> at 20:7-14. “The alu 214 receives instructions from the array controller (not shown) and supplies its output to the PE register file 1061b.” <i>Id.</i> at 49:22-24. <i>See</i> Ex. 1004, ¶¶70-72.
3. The graphics processing system of claim 1, wherein the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device.	<i>See</i> citations to claim element 1.a above. <i>See also</i> Ex. 1004, ¶73.
4. The graphics processing system of claim 1, wherein a single memory device comprises the first portion and the second portion of the at least one memory device.	<i>See</i> citations to claim element 1.a above. <i>See also</i> Ex. 1004, ¶74.
5[pre]. A graphics processing system comprising:	<i>See</i> citations to claim element 1.[pre] above. <i>See also</i> Ex. 1004, ¶75.
[5.a] 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;	<i>See</i> citations to claim element 1.a above. <i>See also</i> Ex. 1004, ¶76.
[5.b] 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	<i>See</i> citations to claim element 1.b above. <i>See also</i> Ex. 1004, ¶77.

Claim	<i>Stuttard</i> (Ex. 1005)
plurality of vertex command threads; and	
[5.c] a plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread.	“As will be explained in greater detail below, the processing core 10 includes a number of control units: thread manager 102, array controller 104, channel controller 108, a binning unit 1069 per block and a microcode store 105. These control units control the operation of a number of processing blocks 106 which perform the graphics processing itself.” Ex. 1005, 8:1-7. “In the example shown in Figure 2, the processing core 10 is provided with eight processing blocks 106. It will be readily appreciated that any number of processing blocks can be provided in a graphics system using this architecture.” <i>Id.</i> at 8:9-13. “Figure 3 shows the processing core in more detail. 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.” <i>Id.</i> at 8:17-27. “Each processing block 106 comprises an array 1061 of processor elements (PEs) and a mathematical expression evaluator (MEE) 1062. As will be described in more detail below, a path 1064 for MEE coefficient feedback is provided from the PE memory, as is an input/output channel 1067. Each processing block includes a binning unit 1069, unit 1068 and a transfer engine 1069 for controlling data transfers to and

Claim	<i>Stuttard</i> (Ex. 1005)
	from the input/output channel under instruction from the channel controller 108.” <i>Id.</i> at 8:29-9:3. <i>See also</i> citations to claim element 2 above; Ex. 1004, ¶¶78-79.
6. The graphics processing system of claim 5, wherein the plurality of command processing engines comprises at least one arithmetic logic unit.	“Figure 11 shows a processor unit 1061 <i>a</i> and PE register file 1061 <i>b</i> which form part of the processing element shown in Figures. 3 and 8. The PE1061 includes an arithmetic logic unit (alu) 214 which is connected to receive data values from a block of 8 bit registers 202, 204, 206, 208(designated R, S, V and P) via multiplexers 210 and 212 (A and B).” Ex. 1005, 47:24-30. <i>See</i> Ex. 1004, ¶¶80-82.
7. The graphics processing system of claim 5, wherein the plurality of command processing engines comprises at least one texture processing engine.	“Increasingly, data processing systems are required to process large amounts of data. In addition, users of such systems are demanding that the speed of data processing is increased. One particular example of the need for high speed processing of massive amounts of data is in the computer graphics field. In computer 15 graphics, large amounts of data are produced that relate to, for example, geometry, texture, and colour of objects and shapes to be displayed on a screen. Users of computer graphics are increasingly demanding more lifelike and faster graphical displays which 20 increases the amount of data to be processed and increases the speed at which the data must be processed.” Ex. 1005, 1:9-22. “The received data is then processed to transform it from the host system reference space to the required screen space. For example, three dimensional geometry, view, lighting and shading etc. is performed to produce data depending upon the chosen viewpoint.” <i>Id.</i> at 21:3-7. <i>See</i> Ex. 1004, ¶¶83-84.

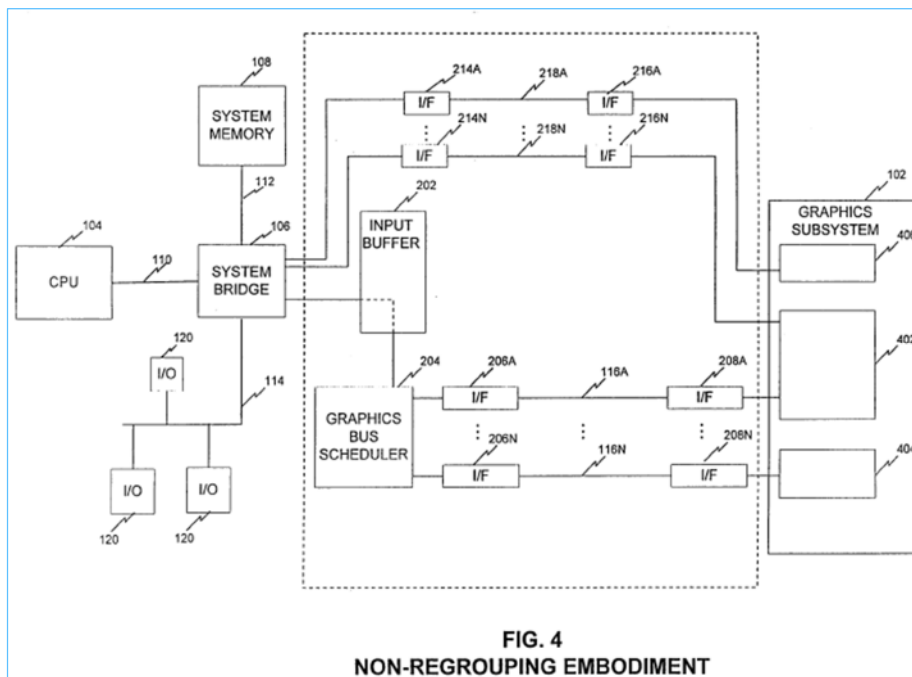
Claim	<i>Stuttard</i> (Ex. 1005)
8. The graphics processing system of claim 5, wherein the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device.	See citations to claim element 1.a above. See also Ex. 1004, ¶85.
9. The graphics processing system of claim 5, wherein a single memory device comprises the first portion and the second portion of the at least one memory device.	See citations to claim element 1.a above. See also Ex. 1004, ¶86.

B. Ground 2: *Stuttard* and *Williams* Render Obvious Claims 1-9

Stuttard in view of *Williams* also renders obvious claims 1-9 of the '053 Patent. The analysis of *Stuttard* above with respect to Ground 1 will not be repeated for the sake of redundancy, but is assumed for this Ground. Thus, the discussion in Ground 2 focuses on *Williams* and is in addition to the analysis of *Stuttard* in Ground 1. 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.

Williams is directed to a graphics processing system with increased command-transfer throughput. Ex. 1006, Abstract. A graphics subsystem or another

subsystem such as a central processing unit (“CPU”) may initiate a transfer of graphics commands. *Id.* at 11:9-15. During the transfer, graphics command may travel from a system memory through a system bridge to the graphics subsystem. *See* Ex. 1004, ¶¶88-89.

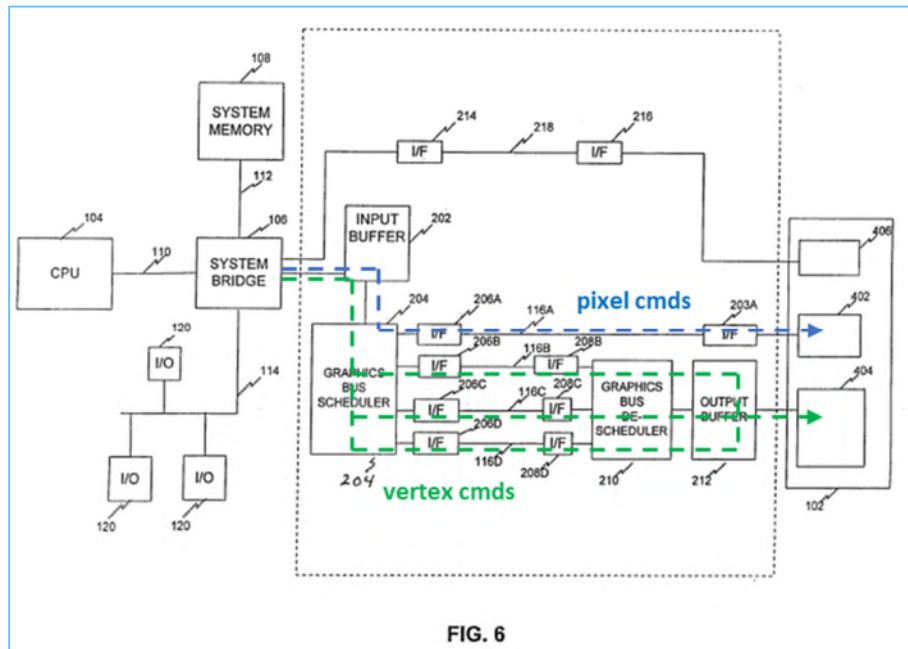


An embodiment of Williams’ graphics system is illustrated in Fig. 4. A system bridge couples the CPU, the system memory, and a plurality of input/output

devices. The system memory stores graphics commands. Ex. 1006 at 11:10-12. The system bridge is also coupled to the graphics system through a plurality of graphic buses, a graphics bus scheduler (“GBS”), and an input buffer. *Id.* at 18:23-25. The input buffer temporarily stores graphics commands during transfer. *Id.* at 11:20-22. The GBS retrieves graphics commands from the input buffer and can assign them to buses according to the command type, e.g., vertex and pixel commands may be assigned to a first and a second bus, respectively. *Id.* at 18:26-

28, 19:4-7. Different assigned buses transfer these different types of commands to the respective functional component of the graphics subsystem. *Id.* at 19:7-11; *see also* Ex. 1004, ¶¶90-91, 96-98.

In this embodiment, the order in which commands are stored in the system memory and/or the input buffer is the order in which the GBS transfers them across the buses. Ex. 1006 at 18:14-18; Ex. 1004, ¶¶90, 99. Because the GBS transfers different types of graphics commands across different buses, it would have been obvious to store different types of graphics commands in different portions of the system memory and/or the input buffer for efficiency of command fetching and transfer, which is *Williams'* objective. *See* Ex. 1006, Abstract (stating the objective of increasing the throughput of graphics command transfer); *see also* Ex. 1004, ¶¶99. For example, 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. *See* Ex. 1004 ¶¶99. Reading or writing consecutive memory addresses can also take advantage of automatic address increment in the hardware, further increasing the efficiency of the operations. *See id.*



In an alternative embodiment, vertex commands are transferred to a graphics bus de-scheduler and an output buffer before being routed to the

graphics subsystem. *Id.* at 23:11-14. In contrast, pixel commands are directly transferred from the input buffer to the graphics subsystem. *Id.* at 23:11-13. Thus, the vertex commands are stored in the output buffer while the pixel commands are stored in the input buffer, and these are two separate devices. *See* Ex. 1004, ¶100.

Motivation to Combine

A POSITA would have been motivated to combine *Stuttard* and *Williams*. Ex. 1004, ¶101.

Both *Stuttard* and *Williams* are directed to the field of graphics processing. Ex. 1005, 4:3-5; Ex. 1006, Abstract. Thus, a POSITA would have been familiar with both references. *See* Ex. 1004, ¶102. Further, the graphics systems in the two references share common components such as a host system with a CPU and a

graphics subsystem. See Ex. 1005, Fig. 1; Ex. 1006, Fig. 1B. In both, the host system provides data and graphics commands to the graphics subsystem. See, e.g., Ex. 1005, 5:18-22; Ex. 1006, 3:15-16. Both involve processing of conventional graphics objects including vertex and pixel commands. Ex. 1005, 1:14-17; Ex. 1006, 6:23-24. Thus, the two systems are compatible and a POSITA would have had a reasonable expectation of success in combining them. See Ex. 1004, ¶102. Further, storing vertex command threads and pixel command threads in two different portions as taught in *Williams* would have involved nothing more than separately storing the respective threads. This would merely apply existing technology with predictable results. See *id.* Moreover, a POSITA would have known, as shown in the graphics processing pipelines shown in these references, vertex and pixel commands are used in distinct stages of graphics processing. See Ex. 1005, 20:21-22:32; Ex. 1006, Fig. 7; see also Ex. 1004, ¶99. Thus, storing vertex commands and pixel commands in separate storage as taught in *Williams* would have been an obvious design choice because of their separation in the processing pipeline. See Ex. 1004, ¶99.

Further, a POSITA would have been motivated to combine them. Both *Stuttard* and *Williams* share the objective of increasing the speed of graphics processing. Ex. 1005, 1:18-22; Ex. 1006, Abstract. *Stuttard* focuses on achieving this objective through an innovative processing array. Ex. 1005, Abstract. *Williams*

is directed to achieving the same objective through increasing the throughput of transferring graphics commands to the graphics subsystem. Ex. 1006, Abstract. A POSITA would have found their teachings complementary because they focus on different aspects to achieve a shared objective. See Ex. 1004, ¶103. The combined system benefits from *Stuttard*'s innovative processing array and *Williams*' high-throughput command-transfer architecture. See *id.* Moreover, a POSITA would have been motivated by their express disclosures of the host-graphics-subsystem interface that *Williams* improved. *Stuttard*'s system uses an AGP bus to connect the graphics subsystem to the host CPU. Ex. 1005, 5:9-11. *Williams* expressly teaches that an AGP graphics bus limits data transfer rate and can be a bottleneck. Ex. 1006, 26:10-15. Thus, a POSITA would have been motivated to apply *Williams*' teaching to achieve a greater throughput and reduce the need for expensive local graphics memory. Ex. 1006, 7:24-8:2; see also Ex. 1004, ¶104. *Williams* also teaches that the system bridge that is part of its system can provide multiple functionalities: (1) functioning as a memory controller; (2) supporting address translation; (3) maintaining cache coherency. Ex. 1006, 5:9-6:2. A POSITA would have recognized these as additional advantages of applying *Williams*' teaching to *Stuttard*'s system. See Ex. 1004, ¶105. Further, a POSITA would have been motivated to separately store vertex and pixel commands as taught in *Williams* because storing the respective command threads in separate portions of a memory device facilitates their access by

the arbiter. For example, a pointer to the respective portions of memory can be easily configured based on which command thread the arbiter is selecting. *See id.* ¶106. For these reasons, a POSITA would have been motivated to combine *Williams*’ command-transfer scheme with *Stuttard*’s graphics-processing array to form an improved graphics-processing system. *See id.* ¶¶101-106.

Accordingly, and as explained by the claim chart below, *Stuttard* in view of *Williams* teaches all limitations of claims 1-9 of the ’053 Patent. *See* Ex. 1004, ¶¶88-120.

Claim	<i>Stuttard</i> (Ex. 1005) in View of <i>Williams</i> (Ex. 1006)
1[pre]. A graphics processing system comprising:	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶93.
[1.a] 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; and	<i>See supra</i> Ground 1. “In one example, the graphics bus scheduler assigns each of the multiple graphics commands to the plurality of busses according to a type of command (e.g., geometry, texture). In another example, the graphics bus scheduler assigns the multiple graphics commands to the plurality of busses in such a manner as to create a pipeline effect across the plurality of busses.” Ex. 1006 at 8:20-24. “In an alternative non-regrouping embodiment, the apparatus of the present invention transfers graphics commands from the system bridge directly to specific functional components of the graphics subsystem. In this embodiment the graphics bus scheduler can assign each of the multiple graphics commands to one of the plurality of busses according to a type of

Claim	<i>Stuttard</i> (Ex. 1005) in View of <i>Williams</i> (Ex. 1006)
	command. Further, in this embodiment the multiple graphics commands are not tagged and are not regrouped.” <i>Id.</i> at 9:8-14. “Input buffer 202, which is coupled to system bridge 106, temporarily stores graphics commands that are to be transferred from system bridge 106 to graphics subsystem 102. Input buffer 202 stores the graphics commands such that their original order (i.e., the order in which the graphics commands were transferred from system bridge 106 to input buffer 202) is known.” <i>Id.</i> at 11:20-24. “FIG. 4 illustrates an alternative embodiment of the present invention in which GBS 204 assigns/sorts graphics commands according to their type, and sends the commands directly to specific functional components of graphics subsystem 102. Thus, in this embodiment there is no need to reorder the graphics commands after they are transferred across the plurality of busses 116A-116N.” <i>Id.</i> at 18:14-18. “GBS 204 receives multiple graphics commands directly from system bridge 106, or from input buffer 202. As in the above embodiment, GBS 204 assigns each of the multiple graphics commands to one of the plurality of busses 116A-116N. However, in this embodiment GBS 204 does not tag the graphics commands because there is no need to reorder the commands after they are transferred across the plurality of busses 116A-116N.” <i>Id.</i> at 18:26-19:3. “GBS 202 preferably assigns and/or sorts the graphics commands according to the type of command. For example, geometry type graphics commands (e.g., vertices, normals, colors) are assigned to a first bus and pixel (e.g., texture)

Claim	<i>Stuttard</i> (Ex. 1005) in View of <i>Williams</i> (Ex. 1006)
	type graphics commands are assigned to a second bus. The plurality of busses 116A-116N transfer the multiple graphics commands between GBS 204 and graphics subsystem 102, such that each of the multiple graphics commands is transferred across its assigned bus to a specific functional component of graphics subsystem 102.” <i>Id.</i> at 19:4-11. “For example, GBS 204 transfers pixel (e.g., texture) related graphics commands across bus 116A directly to an appropriate functional component of graphics subsystem 102, and transfers geometry related graphics commands across busses 116B-116D to GBD 210.” <i>Id.</i> at 23:11-14. <i>See</i> Ex. 1004, ¶¶94-106.
[1.b] 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.	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶107.
2. The graphics processing system of claim 1, further comprising: a command processing engine, coupled to the arbiter, wherein the arbiter is further operable to provide the	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶108.

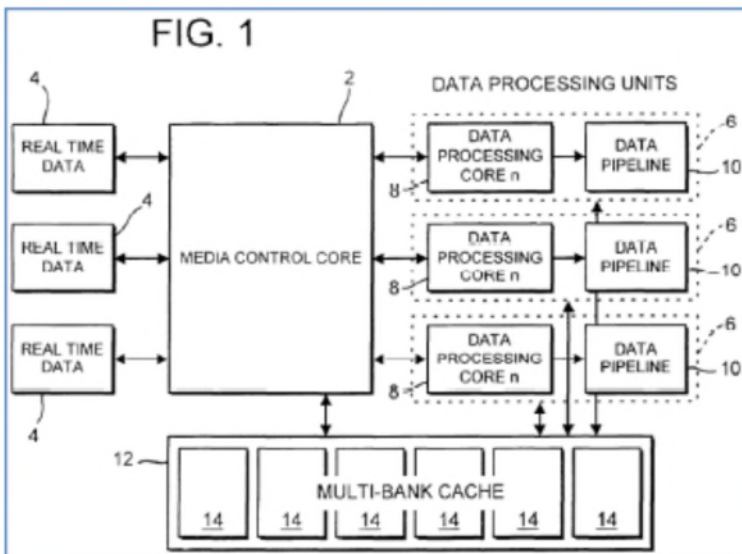
Claim	<i>Stuttard</i> (Ex. 1005) in View of <i>Williams</i> (Ex. 1006)
command thread to the command processing engine.	
3. The graphics processing system of claim 1, wherein the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device.	“For example, GBS 204 transfers pixel (e.g., texture) related graphics commands across bus 116A directly to an appropriate functional component of graphics subsystem 102, and transfers geometry related graphics commands across busses 116B-116D to GBD 210.” <i>Id.</i> at 23:11-14. <i>See</i> Ex. 1004, ¶¶109-111.
4. The graphics processing system of claim 1, wherein a single memory device comprises the first portion and the second portion of the at least one memory device.	<i>See supra</i> claim [1.a]. <i>See</i> Ex. 1004, ¶112.
5[pre]. A graphics processing system comprising:	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶113.
[5.a] 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;	<i>See supra</i> claim [1.a]. <i>See</i> Ex. 1004, ¶114.
[5.b] an arbiter, coupled to the at least one memory device, operable to select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads; and	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶115.

Claim	<i>Stuttard</i> (Ex. 1005) in View of <i>Williams</i> (Ex. 1006)
[5.c] a plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread.	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶116.
6. The graphics processing system of claim 5, wherein the plurality of command processing engines comprises at least one arithmetic logic unit.	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶117.
7. The graphics processing system of claim 5, wherein the plurality of command processing engines comprises at least one texture processing engine.	<i>See supra</i> Ground 1. <i>See</i> Ex. 1004, ¶118.
8. The graphics processing system of claim 5, wherein the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device.	<i>See supra</i> claim 3. <i>See</i> Ex. 1004, ¶119.
9. The graphics processing system of claim 5, wherein a single memory device comprises the first portion and the second portion of the at least one memory device.	<i>See supra</i> claim 4. <i>See</i> Ex. 1004, ¶120.

C. Ground 3: *Williams* and *Whittaker* Render Obvious Claims 1-9

Williams in view of *Whittaker* also renders obvious claims 1-9 of the '053 Patent. Specifically, the combination renders obvious the requirement for the arbiter to select a command thread based on relative priorities. Like Ground 2, Ground 3 assumes the analysis for *Williams* (from Ground 2) and focuses on the additional teachings of *Whittaker*.

Williams does not explicitly disclose that the graphics bus scheduler (“GBS”) selects threads “based on relative priorities” of the threads as required by claim 1. However, this type of priority selection is well known in the art of graphics data processing systems. For example, *Whittaker* discloses a system wherein a thread is selected for processing based on relative priorities. See Ex. 1004, ¶122.



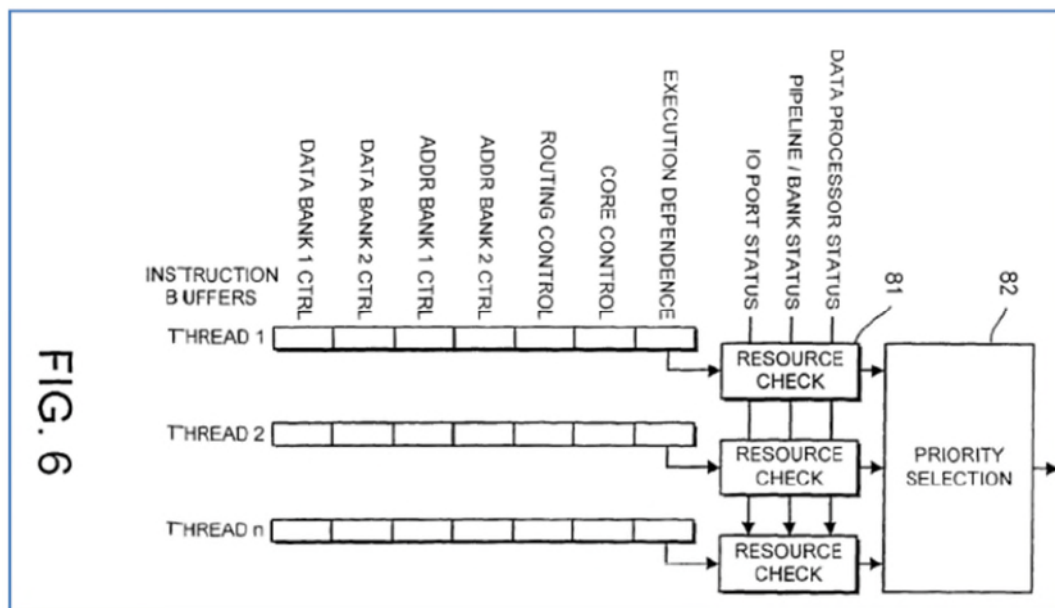
Whittaker is directed to a graphics data processing management system for controlling the execution of multiple threads. See Ex. 1007, Abstract; 1:9-26. Command threads include

those for 3D graphics processing, *e.g.*, pixel processing. *Id.* at 6:42-56. At the center of the system is a Media Control Core (MCC) 2. As shown in Fig. 1, MCC 2

includes a plurality of inputs and output devices 4, data processing units 6, and multi-bank cache 14. *See* Ex. ¶¶ 122-123, 135.

The MCC 2 checks “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.” *Id.* at 3:60-63; *see* Ex. 1004, ¶¶ 123, 135.

This specific resource checking and priority selection of multiple threads is further illustrated in Fig. 6.



The MCC performs this functionality. *Id.* at 3:28-30. As seen, instruction buffers store threads 1, 2, and 3, as well as their various global status information. *Id.* at 8:17-23. The threads and their status information are fed into a resource checker 81, which “ensures that everything required to perform a particular task is in place.” *Id.* at 3:66-67. If all the information required to perform a task is present,

the thread is sent to priority selector 82, which selects for execution the thread with the highest priority. *Id.* 8:28-32; *see also* Ex. 1004, ¶¶136-137. For example, a thread executing real-time data can take precedence over a background processing thread. Ex. 1007 at 8:31-34.

Specifically, in the combination, the priority information of *Whittaker* is provided to *Williams*'s GBS, which, when assigning command threads to buses according to the command type, makes the assignment based on the priority information of the respective threads. *See* Ex. 1006, 18:26-28, 19:4-7 (thread assignment by type); *see also* Ex. 1004, ¶¶133-137.

The Board has already determined that U.S. Patent No. 7,233,335 (“*Moreton*”) in view of *Whittaker* rendered obvious claims 1 and 2 of the ’053 Patent. Ex. 1011 at 75. Like *Williams*, *Moreton* discloses a multithreaded graphics processing system but does not explicitly disclose selecting a thread base on relative priorities of the threads. *Id.* at 58, 68. Because the Patent Owner was able to swear behind *Moreton*, Ground 3 uses analogous art *Williams* and combines *Williams* and *Whittaker* in an analogous manner as *Moreton* and *Whittaker* that the Board has already found to invalidate claims 1 and 2 of the ’053 Patent. *See id.* at 68 (summarizing previous Patent Challenger’s argument).

Motivation to Combine

Both *Williams* and *Whittaker* are directed to the field of 3D graphics processing and rendering. Ex. 1006, 2:7-12; Ex. 1007, 1:9-20. Thus, a POSITA would have been familiar with both references. See Ex. 1004, ¶138. Additionally, they share specific commonalities that would have rendered a combination obvious. First, both involve processing of conventional 3D graphics objects and discuss use of commands related to vertex and pixel processing. Ex. 1006, 6:23-24; Ex. 1007, 6:50-56. Second, *Williams*' GBS and *Whittaker*'s MCC both deal with the issue of parallelism related to multi-thread processing and perform the function of thread selection. Ex. 1006, 18:26-28, 19:4-7; Ex. 1007, 3:28-30. Third, both patents address factors such as resource availability and higher performance when selecting threads for execution. Ex. 1006, Abstract, 12:11-14; Ex. 1007, Abstract, 1:57-63. Thus, the two patents propose different solutions to the same problem of faster 3D graphics processing, and a POSITA would have had a reasonable expectation of success in combining them in the complementary fashion as proposed. See Ex. 1004, ¶138.

Moreover, a POSITA would have been motivated to combine them. Both *Williams* and *Whittaker* share the objective of improving the efficiency of transferring data and commands to the graphics processing subsystem. Ex. 1006, 15:2-3; Ex. 1007, 3:5-8, 4:45-55. Thus, it would have been obvious to one of

ordinary skill in the art to modify *Williams* to include *Whittaker*'s priority-selection function to achieve more efficient data movement and graphics processing, contributing to a commercially significant goal of performance improvement. See Ex. 1004, ¶139. For example, it would have been obvious for the purpose of prioritizing processing of data that requires immediate output such as real time video data. See Ex. 1007, 8:29-34; Ex. 1006, 4:22-29 (disclosing an embodiment with a graphic bus to support real-time data transfer). More specifically, *Williams*' GBS and *Whittaker*'s MCC (via, *e.g.*, the priority mechanism) shared a common objective of achieving a high transfer throughput to the graphics system. See Ex. 1004, ¶139. This common objective of counterpart components would have further motivated a POSITA to apply *Whittaker*'s priority mechanism to *Williams*' GBS to improve transfer throughput. See *id.* ¶139. Further, it would have been obvious to incorporate in *Williams* a priority *selection* function such as the one taught by *Whittaker*, because variation of load among multiple instruction streams can be better and more flexibly balanced to improve the throughput of the processing units. See *id.* ¶139. Because *Williams* and *Moreton* are analogous art, the motivation to combine *Williams* and *Whittaker* is similar to the motivation to combine *Moreton* and *Whittaker* that the Board has already found to provide sufficient rationale for a POSITA to modify a multithreaded graphics processing system like *Moreton* in view of *Whittaker*. See Ex. 1011 at 71-75; see also Ex. 1004, ¶¶ 138-139.

Accordingly, and as explained by the claim chart below, *Williams* in view of *Whittaker* teaches all limitations of claims 1-9 of the '053 Patent. *See* Ex. 1004, ¶¶122-151.

Claim	Williams (Ex. 1006) in View of Whittaker (Ex. 1007)
1[pre]. A graphics processing system comprising:	“An apparatus and method is provided that increases the throughput of graphics commands transferred between a system bridge and a graphics subsystem.” Ex. 1006, Abstract. <i>See</i> Ex. 1004, ¶¶125-131.
[1.a] 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; and	<i>See supra</i> Ground 2 (Ex. 1006 references, <i>i.e.</i>, Ground 2 with respect to <i>Williams</i>). <i>See</i> Ex. 1004, ¶132.
[1.b] 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.	“In one example, the graphics bus scheduler assigns each of the multiple graphics commands to the plurality of busses according to a type of FIG. 4 NON-REGROUPING EMBODIMENT

Claim	<i>Williams</i> (Ex. 1006) in View of <i>Whittaker</i> (Ex. 1007)
	command (e.g., geometry, texture).” Ex. 1006 at 8:20-22. “GBS 204 receives multiple graphics commands directly from system bridge 106, or from input buffer 202.” <i>Id.</i> at 18:26-27. “GBS 202 preferably assigns and/or sorts the graphics commands according to the type of command. For example, geometry type graphics commands (e.g., vertices, normals, colors) are assigned to a first bus and pixel (e.g., texture) type graphics commands are assigned to a second bus. The plurality of busses 116 A- 116N transfer the multiple graphics commands between GBS 204 and graphics subsystem 102, such that each of the multiple graphics commands is transferred across its assigned bus to a specific functional component of graphics subsystem 102.” <i>Id.</i> at 19:4-11; <i>see also id.</i> at 18:15-17 (similar). “In this example, GBS 204 assigns geometry type graphics commands to a first bus 116A and pixel (e.g., texture) type graphics commands to a second bus 116B.” <i>Id.</i> at 19:17-19; <i>see also id.</i> at 34:8-13 (similar). “The resource checking and priority is illustrated fully in FIG. 6. For the three threads illustrated, 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. All 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 the various data processing units. This happens for each possible

Claim	<i>Williams</i> (Ex. 1006) in View of <i>Whittaker</i> (Ex. 1007)
	thread. If it is possible to run that data processing thread then an output is generated to a is priority selector 82. This has information about the priority of each of the data processing threads supported and, as a result, can select for execution the thread with highest priority. For example, a real time data input such a video would be given a high priority and this would take precedence over a background processing operation.” Ex. 1007 at 8:17-34. “The media control core is a fine grained multithreading processing unit which directs data from inputs to data processing cores or to storage and provides data to outputs. It is arranged so that it can switch tasks on every clock cycle. This is achieved by, on every clock cycle checking 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. It could be arranged to commence operation of more than one operation on each clock cycle if sufficient processing power were provided.” <i>Id.</i> at 3:56-65. See Ex. 1004, ¶¶133-139.
2. The graphics processing system of claim 1, further comprising: a command processing engine, coupled to the arbiter, wherein the arbiter is further operable to provide the command thread to the command processing engine.	“GBS 202 preferably assigns and/or sorts the graphics commands according to the type of command. For example, geometry type graphics commands (e.g., vertices, normals, colors) are assigned to a first bus and pixel (e.g., texture) type graphics commands are assigned to a second bus. The plurality of busses 116A-116N transfer the multiple graphics commands between GBS 204 and graphics subsystem 102, such that each of the multiple graphics commands is transferred across its assigned bus to a specific functional

Claim	<i>Williams</i> (Ex. 1006) in View of <i>Whittaker</i> (Ex. 1007)
	component of graphics subsystem 102.” Ex. 1006 at 19:4-11. “After being transferred across the plurality of busses 116A-116N, the multiple graphics commands are provided to specific functional components 402, 404 within graphics subsystem 102. For example, components 402 and 404 can be functional components of graphics subsystem 102 that perform geometry and pixel operations, respectively.” <i>Id.</i> at 22:3-7. See Ex. 1004, ¶140.
3. The graphics processing system of claim 1, wherein the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device.	<i>See supra</i> Ground 2. See Ex. 1004, ¶141.
4. The graphics processing system of claim 1, wherein a single memory device comprises the first portion and the second portion of the at least one memory device.	<i>See supra</i> claim [1.a]. See Ex. 1004, ¶¶142-143.
5[pre]. A graphics processing system comprising:	<i>See supra</i> claim 1[pre]. See Ex. 1004, ¶144.
[5.a] 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;	<i>See supra</i> claim [1.a]. See Ex. 1004, ¶145.

Claim	<i>Williams</i> (Ex. 1006) in View of <i>Whittaker</i> (Ex. 1007)
[5.b] an arbiter, coupled to the at least one memory device, operable to select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads; and	<i>See supra</i> claim [1.b]. <i>See</i> Ex. 1004, ¶146.
[5.c] a plurality of command processing engines, coupled to the arbiter, each operable to receive and process the command thread.	<i>See supra</i> claim 2. <i>See</i> Ex. 1004, ¶147.
6. The graphics processing system of claim 5, wherein the plurality of command processing engines comprises at least one arithmetic logic unit.	“These geometry calculations can be performed by CPU 104, which is well-suited for performing these calculations because it can handle the floating point type operations that are often required. Alternatively, these calculations can be performed by the local processor of graphics subsystem 102.” Ex. 1006 at 2:26-30. “Evaluator 706 approximates curve and surface geometry by evaluating polynomial commands of input values. During the next stage, per-vertex operations and primitive assembly stage 708 processes geometric primitives . . . Vertices are transformed and lit, and primitives are clipped to a viewport in preparation for the rasterization stage 710.” <i>Id.</i> at 24:18-23. “Per-fragment operations stage 712 performs the final operations on graphics data before the data is stored as pixels in frame buffer 714. These final operations can include . . . blending of incoming pixel colors with stored colors, masking, and other logical operations on pixel values.” <i>Id.</i> at 24:27-25:2. <i>See</i> Ex. 1004, ¶148.

Claim	<i>Williams</i> (Ex. 1006) in View of <i>Whittaker</i> (Ex. 1007)
7. The graphics processing system of claim 5, wherein the plurality of command processing engines comprises at least one texture processing engine.	“For example, graphics subsystem 102 processes texture data in order to create lifelike surfaces and shadows within a 3D image.” Ex. 1006 at 3:1-3. <i>See</i> Ex. 1004, ¶149.
8. The graphics processing system of claim 5, wherein the first portion of the at least one memory device comprises a first memory device and the second portion of the at least one memory device comprises a second memory device.	<i>See supra</i> claim 3. <i>See</i> Ex. 1004, ¶150.
9. The graphics processing system of claim 5, wherein a single memory device comprises the first portion and the second portion of the at least one memory device.	<i>See supra</i> claim 4. <i>See</i> Ex. 1004, ¶151.

VIII. CONCLUSION

Substantial, new, and noncumulative technical teachings have been presented for each of claims 1-9. For these reasons and the disclosures identified in the claim charts above, the references above teach and suggest the subject matter of the '053 Patent in such a manner that there is a reasonable likelihood that Petitioner would prevail on at least one claim. Accordingly, Petitioner requests that the Board grant this petition for *inter partes* review.

Respectfully submitted,

/Jeffrey L. Johnson/

Jeffrey L. Johnson
ORRICK, HERRINGTON & SUTCLIFFE LLP
609 Main, 40th Floor
Houston, TX 77002-3106
Telephone: (713) 658-6400
Facsimile: (713) 658-6401
Email: 3J6PTABDocket@orrick.com

Christopher J. Higgins
ORRICK, HERRINGTON & SUTCLIFFE LLP
Columbia Center
1152 15th Street, N.W.
Washington, D.C. 20005-1706
Telephone: (202) 339-8400
Facsimile: (202) 339-8500
Email: OCHPTABDocket@orrick.com

Steve Baik
WHITE HAT LEGAL
P. O. Box 382
San Jose, CA 95002
Telephone: (650) 618-5282
Email: sbaik@whitehat.legal

*Counsel for Petitioner Realtek
Semiconductor Corp.*

CERTIFICATE OF COMPLIANCE

The undersigned certifies that the foregoing PETITION FOR *INTER PARTES* REVIEW complies with the type volume limitation in 37 C.F.R. § 42.24(a)(1)(i). According to the utilized word-processing system's word count, the petition—excluding the caption, table of contents, table of exhibits, mandatory notices, certificate of word count, and certificate of service—contains 13,307 words.

/Jeffrey L. Johnson/
Jeffrey L. 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;

CERTIFICATE OF SERVICE

The undersigned hereby confirms that the foregoing Petition for *Inter Partes* Review and associated exhibits were caused to be served on March 7, 2023, via overnight courier upon the following counsel of record for Patent Owner:

Sterne, Kessler, Goldstein & Fox P.L.L.C.
1100 New York Avenue N.W.
Washington, DC 20005

Courtesy copies of this Petition and accompanying exhibits were also served via electronic mail on Patent Owner's counsel of record in a related district court litigation:

Adam S. Rizk
MINTZ, LEVIN, COHN, FERRIS, GLOSKY, AND POPEO PC
One Financial Center
Boston, MA 02111
Email: AMD-ITC@mintz.com

/Karen Johnson/
Karen Johnson