

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. IPR2023-00687
U.S. Patent No. 8,854,381

**DECLARATION OF JOHN HART REGARDING PATENT OWNER'S
CONTINGENT REVISED MOTION TO AMEND**

U.S. PATENT NO. 8,854,381

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION	1
II. QUALIFICATIONS AND EXPERIENCE.....	1
III. MATERIALS CONSIDERED	4
IV. LEGAL STANDARDS	7
A. Claim Construction.....	7
B. Obviousness.....	8
V. CLAIM CONSTRUCTION	9
VI. BACKGROUND OF TECHNOLOGY, PATENT OVERVIEW, PRIOR ART SUMMARY	10
VII. THE PROPOSED AMENDMENTS DO NOT INTRODUCE NEW SUBJECT MATTER.....	10
A. Substitute Claim 28 Is Supported by the Originally-Filed Application	10
1. 15. A computing system comprising:	10
2. a memory;	11
3. a first processing unit; and	11
4. a bus coupled to the memory and the first processing unit, wherein the first processing unit comprises:.....	11
5. a plurality of engines configured to receive, from a scheduling module associated with a second processing unit, a plurality of tasks and to load state data associated with each of the plurality of tasks; and	11
6. a shader core configured to receive the plurality of tasks from at least one of the plurality of engines and to execute a first task from the plurality of tasks while executing a second task from the plurality of tasks based on respective	

TABLE OF CONTENTS

(Continued)

	Page
state data associated with each of the first and second tasks.....	12
7. 28. The computing system of claim 15, wherein the shader core comprises: multiple processing elements.....	12
8. a first plurality of processing elements of the multiple processing elements configured to execute the first task based on a first set of state data associated with the first task; and	13
9. a second plurality of processing elements of the multiple processing elements configured to execute the second task based on a second set of state data associated with the second task, ...	13
10. wherein the plurality of engines are arranged in parallel with one another;.....	14
11. wherein each of the plurality of tasks is decomposable into two or more related threads; and.....	14
12. wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.....	15
B. Substitute Claim 29 Is Supported by the Originally-Filed Application	26
1. 29. The computing system of claim 15, wherein the plurality of engines comprises a first queue configured to receive the first task and a second queue configured to receive the second task,.....	27
2. wherein the plurality of engines are arranged in parallel with one another,.....	27
3. wherein each engine is configured to receive at least one of the plurality of tasks via a distinct command buffer for each engine,.....	28

TABLE OF CONTENTS

(Continued)

	Page
4. wherein the scheduling module transmits tasks to command buffers for the engines and determines which task to transmit to which engine's command buffer based on a priority level associated with the task,.....	28
5. wherein each of the plurality of tasks is decomposable into two or more related threads, and.....	29
6. wherein each engine is configured to load state data associated with at least one of the plurality of tasks.....	29
VIII. The proposed substitute claims are not unpatentable.....	30
A. Claim 28	31
1. Wilt.....	31
a. wherein the plurality of engines are arranged in parallel with one another;	32
b. wherein each of the plurality of tasks is decomposable into two or more related threads.....	32
c. wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.	36
2. Jiao	36
a. wherein the plurality of engines are arranged in parallel with one another;	37
b. wherein each of the plurality of tasks is decomposable into two or more related threads; and	37
c. wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.	41
B. Claim 29	41

TABLE OF CONTENTS

(Continued)

	Page
1. Wilt.....	41
a. wherein the plurality of engines are arranged in parallel with one another	41
b. wherein each engine is configured to receive at least one of the plurality of tasks via a distinct command buffer for each engine	42
c. wherein the scheduling module transmits tasks to command buffers for the engines and determines which task to transmit to which engine's command buffer based on a priority level associated with the task, ...	43
d. wherein each of the plurality of tasks is decomposable into two or more related threads; and	43
e. wherein each engine is configured to load state data associated with at least one of the plurality of tasks	44

I. INTRODUCTION

1. I have been retained by ATI Technologies UCL (“ATI” or “Patent Owner”), as an independent expert in this proceeding before the Patent Trial and Appeal Board (“PTAB” or “Board”). I understand that the Board has instituted an *Inter Partes* Review (“IPR”) proceeding of U.S. Patent No. 8,854,381 (the “’381 patent”) (Ex. 1001) in IPR2023-00687.

2. I am providing this testimony for consideration by the Patent Trial and Appeal Board in this proceeding, in connection with Patent Owner’s revised motion to amend. I have also provided a declaration in conjunction with the Patent Owner’s Preliminary Response, the Patent Owner Response, and Patent Owner’s first motion to amend.

3. I am being compensated at my standard hourly rate for my services. I have no personal or financial stake or interest in the outcome of the present proceeding, and my compensation is not dependent in any way upon the outcome of this proceeding.

II. QUALIFICATIONS AND EXPERIENCE

4. I have been a professor of computer science, focusing on computer graphics, since 1992. I am currently a tenured Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign, a department consistently ranked in the top-5 by US News and World Report. I am also Director

of the Office for Professional Education of the University of Illinois at Urbana-Champaign.

5. I received a Bachelor of Science degree in Computer Science from Aurora University in 1987, a Master's of Science in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1989, and a Ph.D. in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1991.

6. I worked as an intern in 1998 at the IBM TJ Watson Research Center in Hawthorne, New York, and in 1999 at AT&T Pixel Machines in Holmdel, New Jersey. I worked as a postdoc in 1991-1992 at the Electronic Visualization Laboratory at the University of Illinois at Chicago with funding from the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. I worked as a visiting researcher in the summer of 2007 for the Graphics Research Group of Adobe Systems in Seattle, Washington.

7. From 1997-2000, I designed, simulated and patented graphics hardware for Silicon Reality and the Evans & Sutherland Computer Corp. I have also consulted for visual effects companies (*e.g.*, Kleiser-Walczak, Blue Sky | VIFX), defense contractors (*e.g.*, SAIC, Pratt & Whitney) and medical imaging companies (*e.g.*, Intrinsic Medical Imaging).

8. My research in computer graphics has been supported by Adobe, DARPA, Intel, Nokia, NVIDIA, NSF and Microsoft. I have published over 75 reviewed papers on computer graphics, including papers in the most rigorous and prestigious venues, including the SIGGRAPH Annual Conference Proceedings, the ACM Transactions on Graphics, and the IEEE Transactions on Visualization and Computer Graphics. From 2002-2008, I was the longest-serving Editor-in-Chief of ACM Transactions on Graphics, the top journal in the field of computer graphics.

9. I was an Executive Producer for the ACM SIGGRAPH documentary “The Story of Computer Graphics” in 1999. I served from 1996-2000 on the ACM SIGGRAPH Executive Committee. I served as the founding computer graphics area editor of the ACM Books series, and oversaw the publication of “The VR Book: Human Centered Design for Virtual Reality.”

10. I co-authored the book “Real-Time Shading” in 2002 on the implementation of procedural shaders on graphics processors became fully user programmable GPU’s. I contributed two chapters to the book “Modeling and Texturing: A Procedural Approach” in 2002, including a chapter on procedural texturing using rasterization and texturing operations.

11. I am being compensated for my time working on this matter at my usual rate of \$575 per hour. My compensation does not depend on the outcome of this litigation.

12. I believe I meet the criteria for a POSITA as held in the Institution Decision in this matter, and any other appropriate definition of a POSITA for this patent.

13. Additional details about my employment history, fields of expertise, and publications are further described in my curriculum vitae, which is attached as Exhibit A to my declaration in support of the Patent Owner’s Preliminary Response (Ex. 2001).

III. MATERIALS CONSIDERED

14. In forming my opinion I have reviewed the proposed amended claim language that counsel provided to me, listed below, and I have relied on at least the following documents, in addition to the pleadings, such as the petition, the institution decision, etc., and any other documents cited herein:

Exhibit	Description
EX1001	U.S. Patent 8,854,381 to Lefebvre et al.
EX1002	Prosecution History of U.S. Patent 8,854,381
EX1004	U.S. Patent No. 8,151,095 to Wilt (“Wilt”)
EX1005	U.S. Patent No. 8,319,774 to Jiao <i>et al.</i> (“Jiao”)
EX1006	PCT Pub. No. WO00/62182A2 to Stuttard <i>et al.</i> (“Stuttard”)

Underline indicates additions to the issued claims. Double underline indicates additions to the issued claims that are new in this Revised Motion to Amend.

Current Claim	Proposed Substitute Claim
15. A computing system comprising: a memory; a first processing unit; and a bus coupled to the memory and the first processing unit, wherein the first processing unit comprises: a plurality of engines configured to receive, from a scheduling module associated with a second processing unit, a plurality of tasks and to load state data associated with each of the plurality of tasks; and a shader core configured to receive the plurality of tasks from at least one of the plurality of engines and to execute a first task from the plurality of tasks while executing a second task from the plurality of tasks based on respective state data associated with each of the first and second tasks.	No substitute claim proposed.
16. The computing system of claim 15, wherein the shader core comprises: a first plurality of processing elements configured to execute the first task based on a first set of state data associated with the first task; and a second plurality of processing elements configured to execute the second task based on a second set of state data associated with the second task.	28. The computing system of claim 15, wherein the shader core comprises: <u>multiple processing elements</u>; a first plurality of processing elements <u>of the multiple processing elements</u> configured to execute the first task based on a first set of state data associated with the first task; and a second plurality of processing elements <u>of the multiple processing elements</u> configured to execute the

Current Claim	Proposed Substitute Claim
	second task based on a second set of state data associated with the second task, <u>wherein the plurality of engines are arranged in parallel with one another;</u> <u>wherein each of the plurality of tasks is decomposable into two or more related threads; and</u> <u>wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.</u>
17. The computing system of claim 15, wherein the plurality of engines comprises a first queue configured to receive the first task and a second queue configured to receive the second task.	29. The computing system of claim 15, wherein the plurality of engines comprises a first queue configured to receive the first task and a second queue configured to receive the second task, <u>wherein the plurality of engines are arranged in parallel with one another; wherein each engine is configured to receive at least one of the plurality of tasks via a distinct command buffer for each engine; wherein the scheduling module transmits tasks to command buffers for the engines and determines which task to transmit to which engine's command buffer based on a priority level associated with the task, wherein each of the plurality of tasks is decomposable into two or more related threads; and wherein each engine is configured to load state data associated with at least one of the plurality of tasks.</u>

Current Claim	Proposed Substitute Claim
18. The computing system of claim 15, wherein the first task comprises a low-latency task and the second task comprises a regular-latency task.	No substitute claim proposed.
19. The computing system of claim 15, wherein the first task comprises a graphics-processing task and the second task comprises a general-compute task.	No substitute claim proposed.
20. The computing system of claim 15, wherein the shader core is configured to receive the plurality of tasks based on a priority associated with each of the plurality of tasks.	No substitute claim proposed.

IV. LEGAL STANDARDS

15. In preparing and expressing my opinions and considering the subject matter of the '381 patent, I am relying on certain basic legal principles that counsel have explained to me. These principles are discussed below.

A. Claim Construction

16. I understand that the first step in determining the validity of a claim is for the claim to be properly construed. I have been further advised that, in an *inter partes* review proceeding, the claims of a patent are typically given their ordinary and customary meaning as would be understood by one of ordinary skill in the art in the context of the claims and the patent disclosure (specification).

17. I further understand that an inventor can provide special definitions for specific claim term(s) within the patent. However, any special definition for a claim term must be set forth in the specification with reasonable clarity and precision.

B. Obviousness

18. I have been informed that a claim may be invalid as obvious if the subject matter described by the claim as a whole would have been obvious to a POSITA at the time the claimed invention was made. I understand that the standard for obviousness in an *inter partes* review proceeding is by a preponderance of the evidence.

19. I have also been informed that a determination of obviousness involves an analysis of the scope and content of the prior art, the similarities between the claimed invention and the prior art, and the level of ordinary skill in the art. I have been informed and understand that a prior art reference should be viewed as a whole.

20. I have been informed that in considering whether an invention for a claimed combination would have been obvious, I may assess whether there are apparent reasons to combine known elements in the prior art in the manner claimed in view of interrelated teachings of multiple prior art references, the effects of demands known to the design community or present in the market place, and/or the background knowledge possessed by a POSITA. I also understand that other

principles may be relied on in evaluating whether a claimed invention would have been obvious.

21. I have been informed that, in making a determination as to whether or not the claimed invention would have been obvious to a POSITA, one may consider certain objective indicators of non-obviousness if they are present, such as: commercial success of product(s) practicing the claimed invention; long-felt but unsolved need; teaching away; unexpected results; copying; and praise by others in the field. I understand that for such objective evidence to be relevant to the non-obviousness of a claim, however, there must be a causal relationship (called a “nexus”) between the claim and the evidence. I also understand that this nexus must be based on a novel element of the claim rather than something available in the prior art.

22. I have also been informed and understand that when considering the obviousness of a patent claim, one should consider whether a reason or motivation existed for combining the elements of the references in the manner claimed, and that the prior art must create a reasonable expectation of success in producing the claimed subject matter.

V. CLAIM CONSTRUCTION

23. In forming my opinions in this matter, I have applied the constructions as found by the Board in the Institution Decision. That is, I have applied the meaning

of “associated with” such that a scheduling module in the first processing unit with the engines is not associated with a second processing unit, and I have given the rest of the claim terms their plain meaning. ID at 9-10. My opinions in this declaration do not depend on any particular construction, although I reserve the right to address any arguments or holdings that are made regarding constructions.

VI. BACKGROUND OF TECHNOLOGY, PATENT OVERVIEW, PRIOR ART SUMMARY

24. My declarations in support of the Patent Owner’s Preliminary Response (Ex. 2001) and Patent Owner’s Response contain opinions regarding the background of the technology, an overview of the ’381 patent, and a summary of the closest-known prior art.

VII. THE PROPOSED AMENDMENTS DO NOT INTRODUCE NEW SUBJECT MATTER

A. Substitute Claim 28 Is Supported by the Originally-Filed Application

1. 15. A computing system comprising:

25. Claim 15, from which claim 28 depends, recites “[a] computing system,” which is disclosed at least in Paragraphs 2, 3, 6, 13-17, 33-42, and Figs. 1-5, *e.g.*, “computing system 100,” of the originally-filed application, among other places. Ex. 1002 at 6-35.

2. a memory;

26. This limitation is disclosed at least in Paragraphs 15, 37-38, 56, Fig. 1 (items 104, 106), and Fig. 2 (system memory) of the originally-filed application, among other places. Ex. 1002 at 6-35.

3. a first processing unit; and

27. This limitation is disclosed at least in Paragraphs 13, 15, 33-45 (GPU 110) Fig. 1 (GPU 110), and Fig. 2 of the originally-filed application, among other places. Ex. 1002 at 6-35.

4. a bus coupled to the memory and the first processing unit, wherein the first processing unit comprises:

28. This limitation is disclosed at least in Paragraphs 15, 37-38, and Fig. 1 (bus 114) of the originally-filed application, among other places. Ex. 1002 at 6-35.

5. a plurality of engines configured to receive, from a scheduling module associated with a second processing unit, a plurality of tasks and to load state data associated with each of the plurality of tasks; and

29. This limitation is disclosed at least in the Abstract, Paragraphs 13, 15, 16, 27-31 (virtual engines), 41, 44, 47, 58-64, Fig. 2 (engines 202), Fig. 3A (engines 312 are a “plurality of engines”; OS Scheduler 310 is a “scheduling module”; application 302 runs on CPU 102, which is a “second processing unit”), Fig. 3B (engines 332 are a “plurality of engines”; OS Scheduler 310 is a “scheduling module” and runs on CPU 102, which is a “second processing unit”), and Fig. 4 (engines 432 are a “plurality of engines”; OS Scheduler 310 is a “scheduling

module” and runs on CPU 102, which is a “second processing unit”) of the originally-filed application, among other places. Ex. 1002 at 6-35. A POSITA would know that the OS Scheduler, *i.e.*, the Operating System Scheduler, runs on the CPU, which runs the Operating System, and thus the OS Scheduler is within the CPU.

6. **a shader core configured to receive the plurality of tasks from at least one of the plurality of engines and to execute a first task from the plurality of tasks while executing a second task from the plurality of tasks based on respective state data associated with each of the first and second tasks.**

30. This limitation is disclosed at least in Paragraphs 13, 15, 27-32, 35, 41-42, 50-54, 58-64, and Figs. 2-4 of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, paragraphs 27, 30, 41, 50-54, and 58-64 describe the ability of the shader core to execute a first task from the plurality of tasks while executing a second task from the plurality of tasks based on respective state data associated with each of the first and second tasks.

7. **28. The computing system of claim 15, wherein the shader core comprises: multiple processing elements**

31. This limitation is disclosed at least in Paragraphs 30, 50-54, and Fig. 2 (depicting processing elements 220 of shader core 214) of the originally-filed application, among other places. Ex. 1002 at 6-35.

- 8. a first plurality of processing elements of the multiple processing elements configured to execute the first task based on a first set of state data associated with the first task; and**

32. This limitation is disclosed at least in Paragraphs 30, 50-54, and Fig. 2 (depicting processing elements 220) of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, Paragraph 30 explains that “resources of the shader core are partitioned in space and/or time.” The specification further states that “[t]o achieve spatial partitioning, for example, a first (e.g., low-latency) task from a first virtual engine is issued to a first subset of processing elements (SIMDs) of the shader core, and a second (e.g., standard graphics) task from a second virtual engine is issued to a second subset of processing elements (SIMDs) of the shader core.” Paragraphs 8 and 16, among others, explain that such execution is based on state data associated with each of the plurality of tasks.

- 9. a second plurality of processing elements of the multiple processing elements configured to execute the second task based on a second set of state data associated with the second task,**

33. This limitation is disclosed at least in 30, 50-54, and Fig. 2 (depicting processing elements 220) of the originally-filed application, among other places. Ex. 1002 at 6-35, 50-54. The citations and explanation in the previous element apply here too.

10. wherein the plurality of engines are arranged in parallel with one another;

34. This limitation is disclosed at least in the Abstract, Paragraphs 13, 15, 29, and Fig. 2 of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, paragraph 13 states that “The plurality of virtual engines is configured to (i) receive, from an operating system (OS), a plurality of tasks substantially in parallel with each other.” Further, paragraph 29 states that “[t]asks from each virtual engine are then issued to a single shader core substantially in parallel with each other.” Moreover, in Figures 2 (elements 202), 3A (elements 312), 3B (elements 312), and 4 (elements 432) the engines are depicted as being arranged in parallel with one another.

11. wherein each of the plurality of tasks is decomposable into two or more related threads; and

35. This limitation is disclosed at least in Paragraphs 5-8 of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, paragraph 5 states that “[t]o perform a single graphics-processing task, a GPU may execute a plurality of different threads (sequence of instructions).” Paragraph 6 similarly refers to “the various threads (e.g., shader programs) of a single graphics-processing task.” Paragraph 8 also refers to “the threads of the first task.” The fact that the threads are “of” a task, for example, indicates that the task is decomposable into those threads and that they are related. Dr. Pedram’s deposition testimony

demonstrates the same understanding, that in general tasks are decomposable into related threads. Ex. 2007 at 13:5-25:4, 29:22-31:4. To the extent it is argued that the '381 patent specification leaves open the possibility that a task may decompose into only one thread, I disagree for the reasons set forth in my declaration filed in conjunction with the Patent Owner Response. However, it does not add new matter to claim a clearly-disclosed embodiment as done here.

12. wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.

36. This limitation is disclosed at least in Paragraphs 27-32, 35, 50-54, and Fig. 2 (depicting processing elements 220) of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, Paragraph 35 states “GPU 110 includes a plurality of virtual engines that share resources of a single shader core.” Paragraph 30 explains that “resources of the shader core are partitioned in space and/or time.” The specification further states that “[t]o achieve temporal partitioning, for example, the first and second tasks share a percentage of time of the processing elements (SIMDs) of the shader core.” Paragraph 43 explains, for example, that the engines can process different types of tasks, and combined these all indicate that one way to execute tasks simultaneously in the patented invention is for the processing elements of the shader core to receive and execute tasks from one engine at one time, and to receive and execute tasks from another engine at another time, and that over time

the processing elements will receive tasks from each engine. Notably, the specification explains that this functionality does not conflict with spatial partitioning because the “resources of the shader core are partitioned in space and/or time.” Ex. 1002 at 12 (emphasis added). This could be simultaneous partitioning of both types, *i.e.*, a hybrid approach in which some subset of elements employ temporal partitioning and some employ spatial partitioning, or the partitioning could be the same across all elements such that all elements are first partitioned in time and then in space, or vice versa.

37. For example, one configuration that is disclosed to a POSITA is that in temporal processing, at time 1, all processing blocks are devoted to receiving real-time low-latency tasks from the engine that handles those tasks:

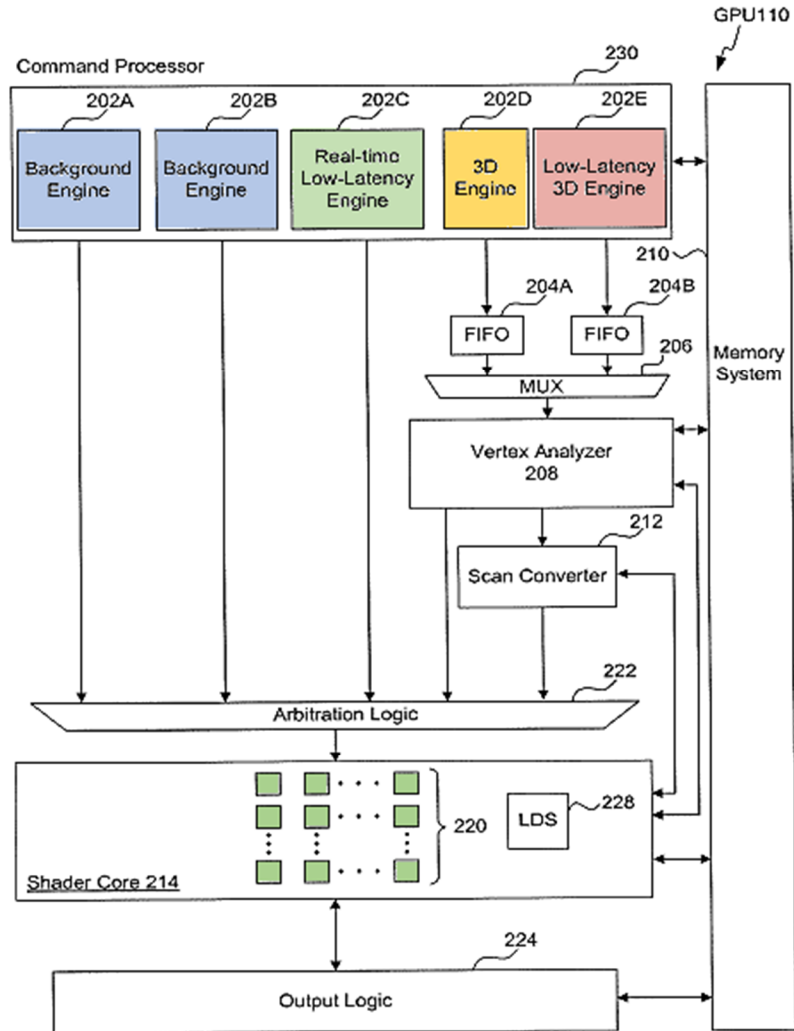


FIG. 2

1022173v1/2

Id.

38. Then at a second time all processing elements can be devoted to receiving tasks from the 3D engine:

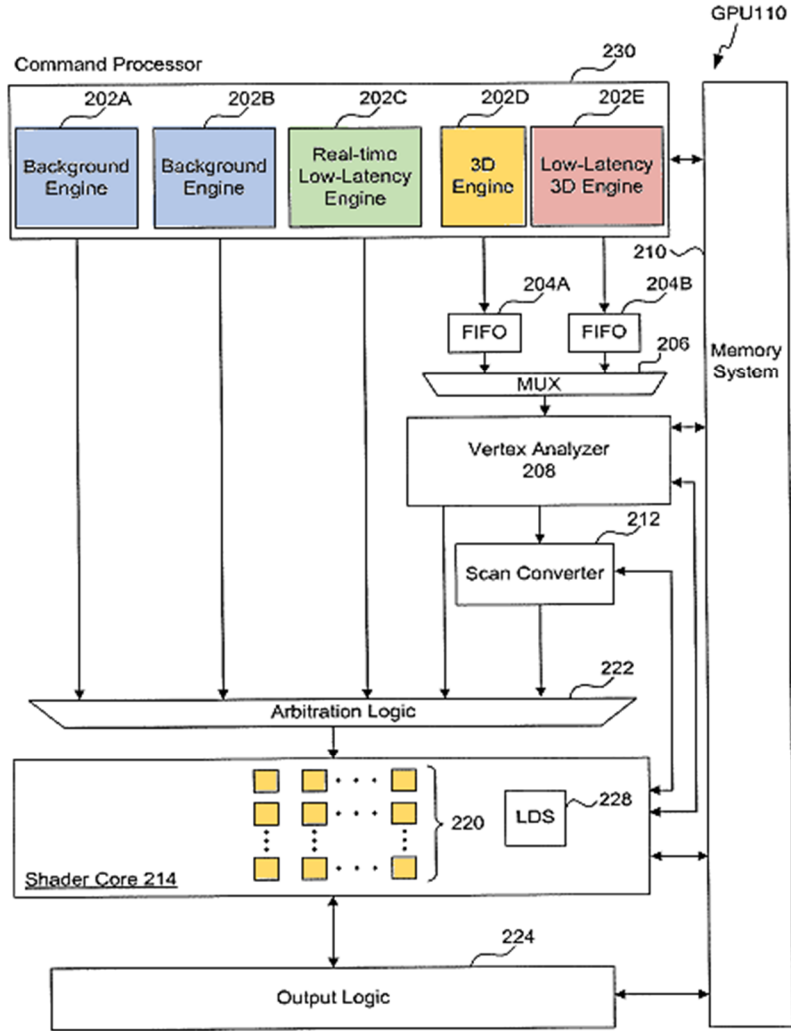


FIG. 2

1022173v1/2

Id.

39. And then at a third time, all processing elements can be receiving tasks from the low-latency 3D engine:

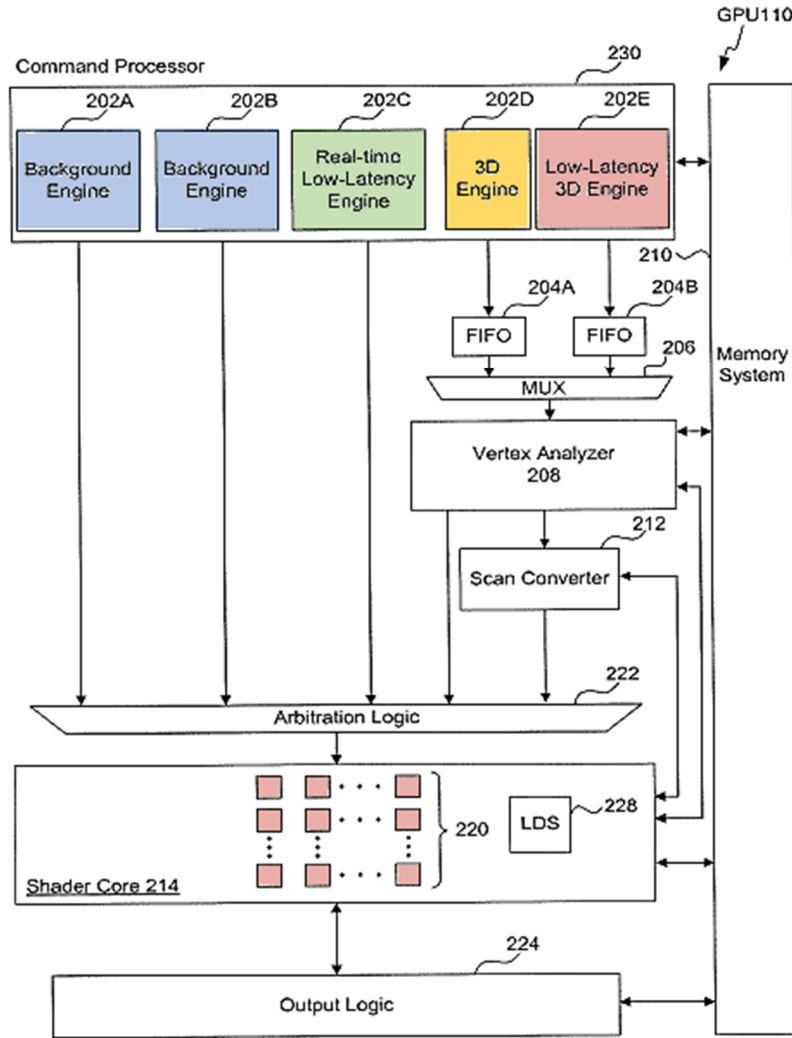


FIG. 2

1022173v1/2

Id. Thus, each processing element is configured to receive tasks from each engine.

40. Of course, other combinations are disclosed to a POSITA as well, in which some processing elements follow the temporal partitioning approach and some follow the spatial portioning approach, in which certain processing elements process certain types of tasks.

41. For example, at time 1, the three depicted processing elements on the right (and intermediate elements not shown) follow temporal portioning, such that over time they receive tasks from all engines, and the six processing elements depicted on the left (and intermediate elements not shown) follow spatial partitioning such that they are configured to receive only tasks from a given engine.

Time 1:

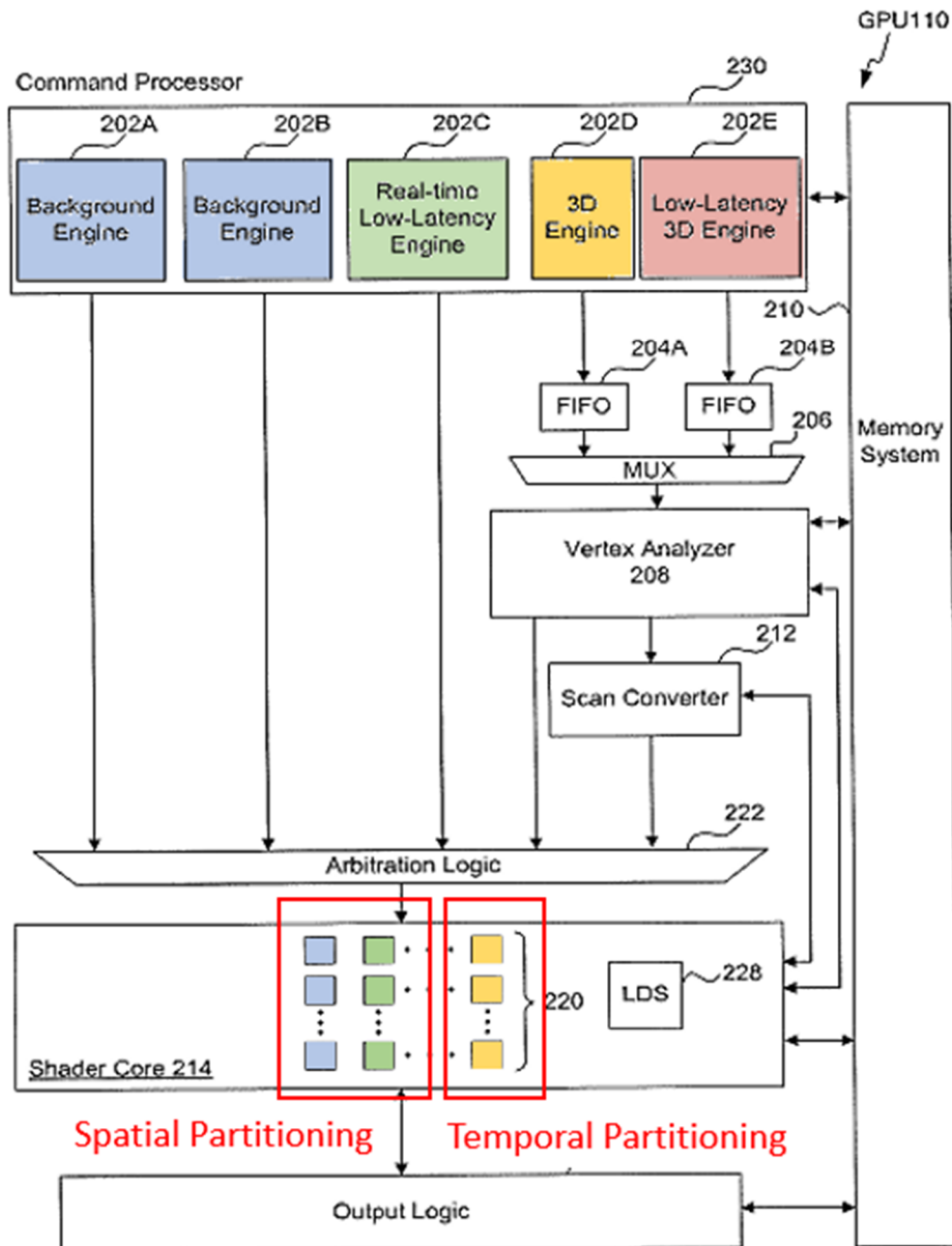


FIG. 2

1022173v1/2

Time 2:

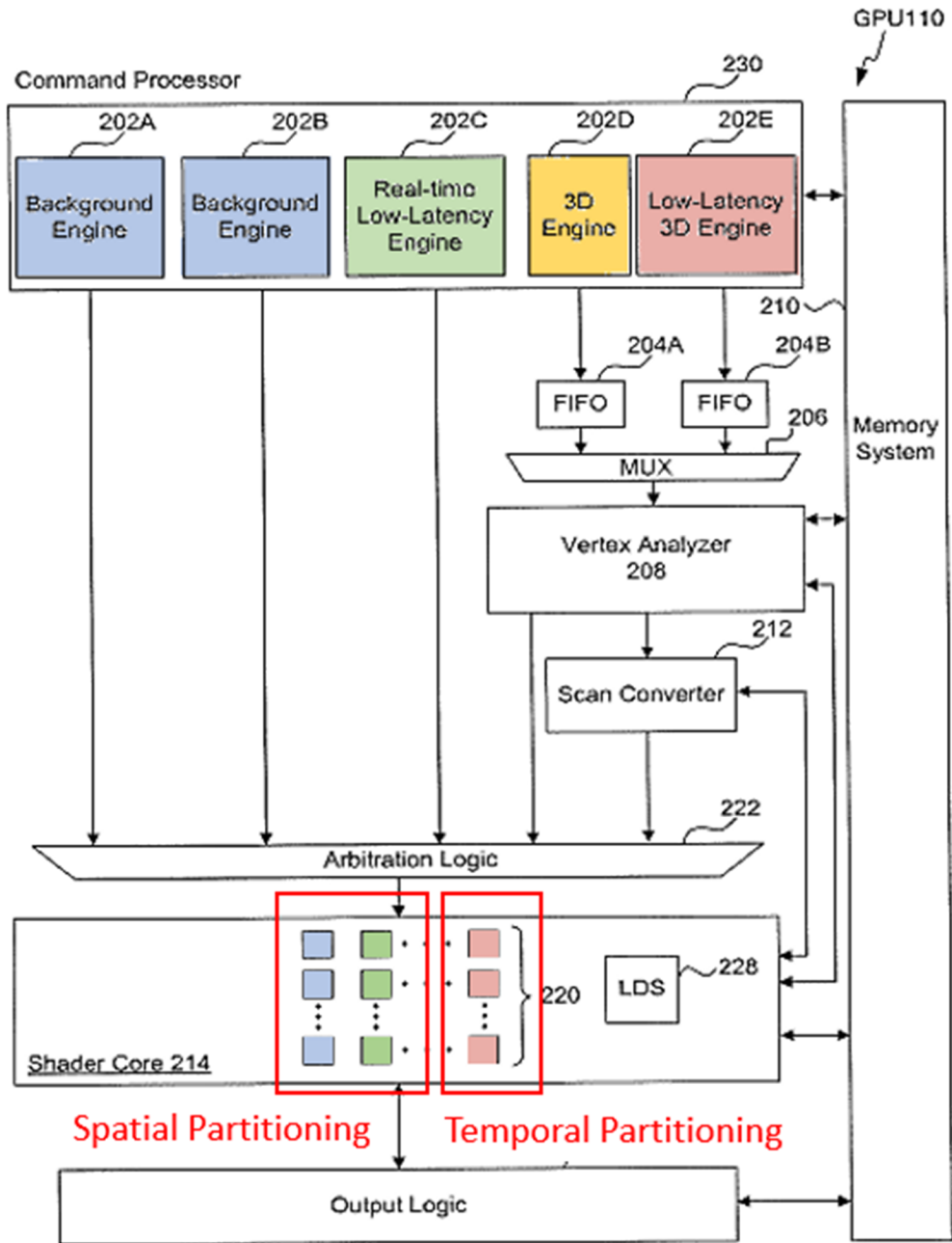


FIG. 2

1022173v1/2

Time 3:

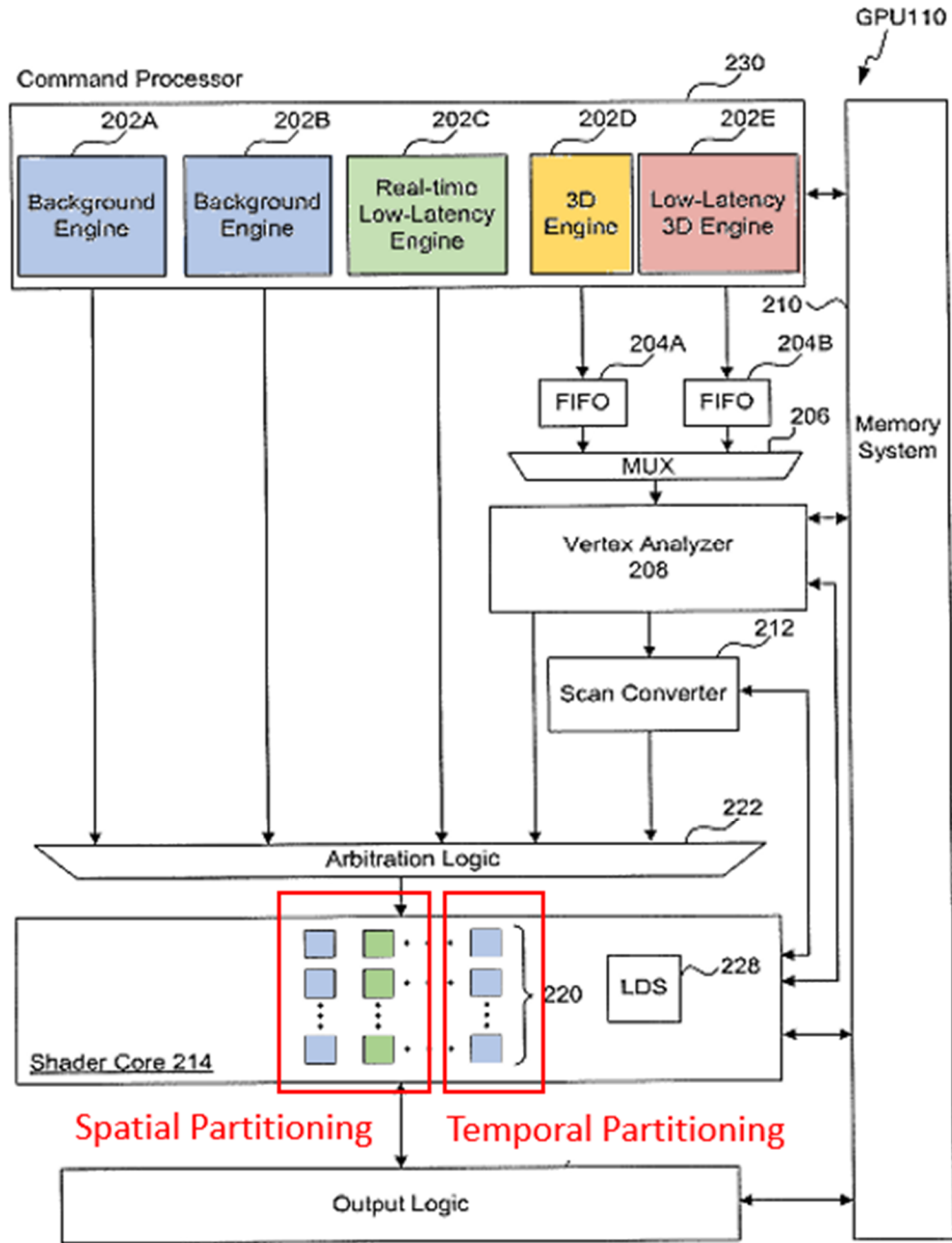


FIG. 2

1022173v1/2

Time 4:

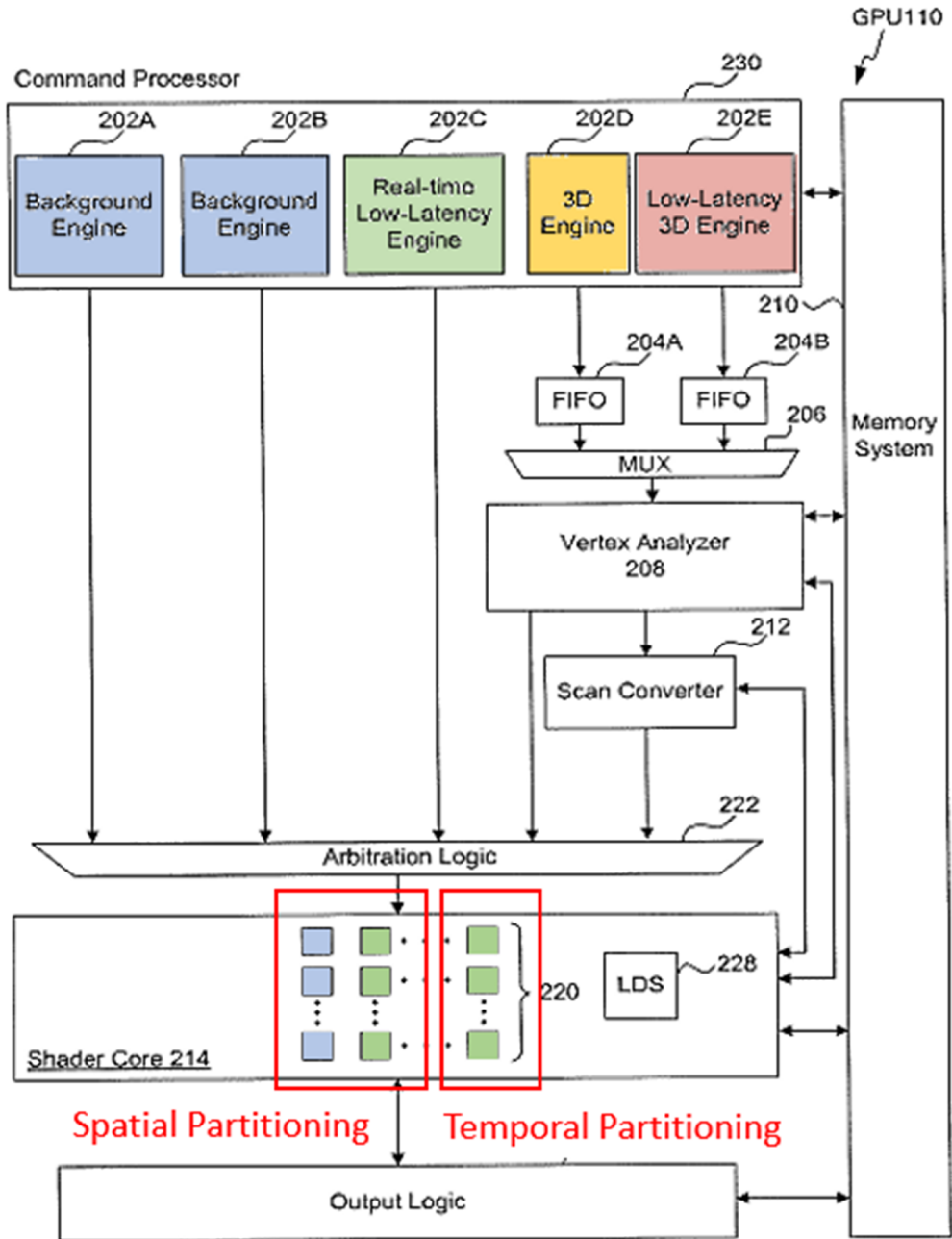


FIG. 2

1022173v1/2

Id. Thus, the processing elements in the right column each receive tasks from each engine.

42. The Preliminary Guidance held that this element does not have support in the originally-filed application because “the ’134 application does not appear to specify whether one processing element may receive an entire task, or whether plural processing elements receive one task and share pieces of the task between themselves.” PG at 8. Patent Owner believes both are disclosed to a POSITA. Regardless, the claims have been further amended to address the Board’s concerns, now stating “at least one processing element is configured to receive at least partial tasks from each engine,” to address the hypothetical possibility that only plural processing elements may ever receive a full task, and an individual processing element may only ever receive part of a task, *i.e.*, a thread or an instruction decomposed from the claimed task.

43. The Preliminary Guidance also held that “the ’134 application describes shader core 214 and processing elements receiving tasks from arbitration logic 222 or scheduling module 434 not engines.” PG at 8. To the extent this is implying that there is no specification support for the shader core and processing elements receiving tasks because they must be directly received, that is contradicted by the holding in the Preliminary Guidance that “the ’381 patent contemplates processing elements that receive tasks from engines directly, or indirectly—through arbitration logic 222 (and also, possibly, through MUX 206, vertex analyzer 208, and scan converter 212).” PG at 31; *see also* Institution Decision at 17 (holding the

claims cover indirect reception). If the holding was merely that it is unclear how the arbitration logic and other components distribute tasks or parts of tasks, that is irrelevant. The temporal partitioning configuration and the spatial and temporal hybrid portioning configuration both demonstrate that a processing element receives and executes tasks (or in the Board's view, potentially only partial tasks), from each engine, regardless of the intermediate distribution steps.

44. The Preliminary Guidance also agreed with Petitioner's argument that the patentee had "no intended configuration at all." PG at 9. That is incorrect. The temporal partitioning configuration and the spatial and temporal hybrid portioning configuration are specifically-described configurations with enough detail to enable a POSITA to practice those configurations. Merely as one example, when all resources (*i.e.*, all processing elements) are configured to process tasks from the same engine at a given time, as in the temporal portioning configuration, all processing elements (and therefore at least one) are configured to receive and execute tasks or partial tasks from all engines, over time. Nothing more is required for written description.

B. Substitute Claim 29 Is Supported by the Originally-Filed Application

45. Claim 15 has support in the originally-filed application, as explained in the previous section.

1. **29. The computing system of claim 15, wherein the plurality of engines comprises a first queue configured to receive the first task and a second queue configured to receive the second task,**

46. This limitation is disclosed at least in Paragraphs 30, 50-54, and Fig. 2 (depicting processing elements 220) of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, as Paragraphs 30 and 50-54 explain, the tasks are received by the engines and then “issued” from the engines to the shader core. As a POSITA would know, the engines would or could use a queue to receive, hold, and then issue the tasks. Claim 3 of the original application also discloses this limitation. Ex. 1002 at 25.

2. **wherein the plurality of engines are arranged in parallel with one another,**

47. This limitation is disclosed at least in the Abstract, Paragraphs 13, 15, 29, and Fig. 2 of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, paragraph 13 states that “The plurality of virtual engines is configured to (i) receive, from an operating system (OS), a plurality of tasks substantially in parallel with each other.” Further, paragraph 29 states that “[t]asks from each virtual engine are then issued to a single shader core substantially in parallel with each other.” Moreover, in Figures 2 (elements 202), 3A (elements 312), 3B (elements 312), and 4 (elements 432) the engines are depicted as being arranged in parallel with one another.

- 3. wherein each engine is configured to receive at least one of the plurality of tasks via a distinct command buffer for each engine,**

48. This limitation is disclosed at least in the Abstract, Paragraphs 7, 29, 39, 43, 63-64, and Fig. 4 of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, paragraph 64 states that “GPU 110 includes a plurality of virtual engines 432A-N, each configured to service one of command buffers 420A-N,” which is depicted in Fig. 4, in which each engine receives tasks via a distinct command buffer. Petitioner’s expert, Dr. Pedram, admitted that is what Fig. 4 depicts. Ex. 2007 at 81:2-5.

- 4. wherein the scheduling module transmits tasks to command buffers for the engines and determines which task to transmit to which engine’s command buffer based on a priority level associated with the task,**

49. This limitation is disclosed at least paragraphs 17, 44, 58-60, 63-64 of Ex. 1002, *e.g.*: “Each task 330 includes a priority type. OS scheduler 310 receives tasks 330 and issues them to different virtual engines of GPU 110 . . . GPU 110 includes a scheduling module that schedules the tasks to be executed on shader core 214 based, at least in part, on the priority type specified by the application.” *Id.* ¶¶ 59-60. Further: “OS scheduler 310 provides each task 404 to one of a plurality of command buffers 420A-N based on the priority type.” *Id.* ¶ 63.

5. wherein each of the plurality of tasks is decomposable into two or more related threads, and

50. This limitation is disclosed for the same reasons as in claim 28 above.

6. wherein each engine is configured to load state data associated with at least one of the plurality of tasks.

51. This limitation is disclosed at least in the Abstract, Paragraphs 13, 16, 27, 53, and Figs. 2, 3A, 3B, and 4 of the originally-filed application, among other places. Ex. 1002 at 6-35. For example, paragraph 14 states that “[t]he plurality of virtual engines is configured to (i) receive, from an operating system (OS), a plurality of tasks substantially in parallel with each other and (ii) load a set of state data associated with each of the plurality of tasks.” As a POSITA would understand this statement is intended to convey that each engine loads the state data associated with the task it receives and transmits to the shader core. For example, that is what is shown in Fig. 4. In Fig. 4 (and other figures), each engine operates independently, to receive tasks, load state data, etc. A serialized sort of design where one engine receives all tasks, another engine loads all data, etc., is not the approach of Fig. 4:

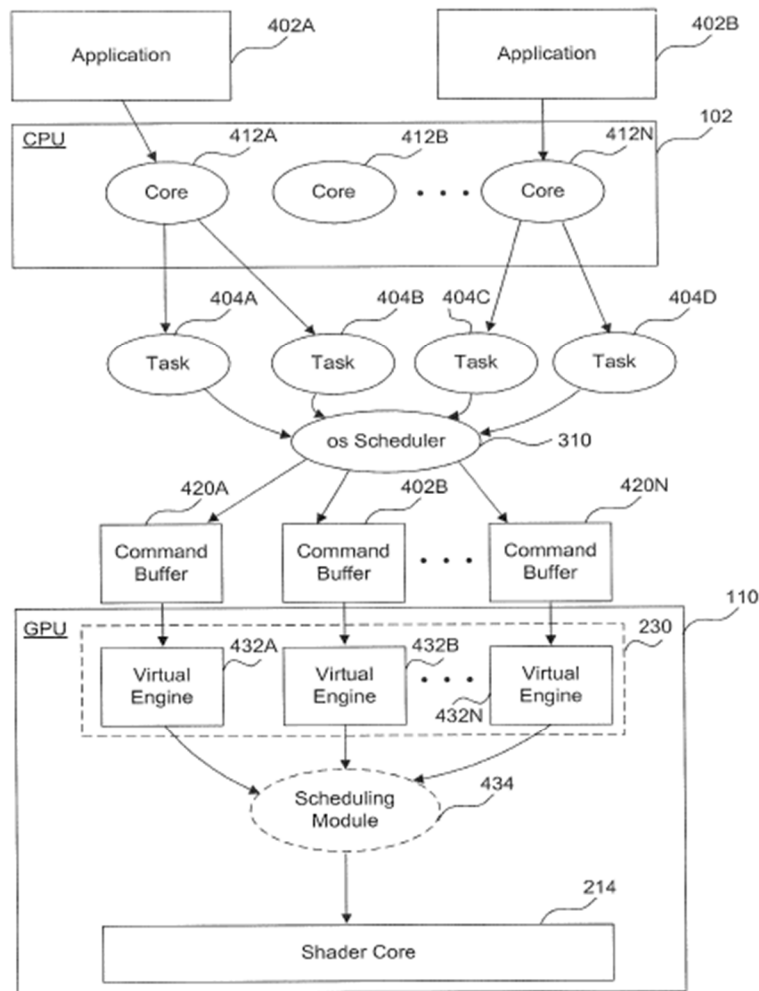


FIG. 4

52. At a minimum, a POSITA would know that was one disclosed design based.

VIII. THE PROPOSED SUBSTITUTE CLAIMS ARE NOT UNPATENTABLE

53. Petitioner has alleged invalidity of the claims in the prior motion to amend on the basis of Wilt (claims 28 and 29) and Jiao (claim 28 only). Petitioner

does not allege that Stuttard invalidates the claims. Patent Owner's revised amendment solely adds limitations to the prior amendment.

54. The below arguments are directed only to the additional claim limitations provided in the contingent amendments. The grounds further fail to invalidate the current claims for reasons explained in my declaration filed in conjunction with the Patent Owner's Response. Those arguments are not repeated here.

55. I reserve the right to respond with additional arguments as the schedule permits.

A. Claim 28

1. Wilt

56. Petitioner has alleged two theories for what constitute the claimed plurality of engines and the claimed shader core. Under what I will call the "Single PPU Theory," the work distribution unit 210, the front-end unit 212, and the core interface unit 303 of a single PPU are each an engine, such that they are the plurality of engines, and a single PPU is the shader core. The Institution Decision explained that another theory could be that the cores 208 within one PPU together are the shader core.

57. Under Petitioner's second theory, the "Multiple PPU Theory," the work distribution unit 210, the front-end unit 212, and the core interface unit 303 of a

single PPU are together a single engine, and that combination of components within each PPU taken together are the claimed plurality of engines. The Petitioner argued under this theory that all PPUs taken together are the claimed shader core. The Institution Decision explained that another theory could be that all cores 208 within all PPUs together are the shader core.

58. Under any theory, Wilt fails to disclose claim 28.

a. wherein the plurality of engines are arranged in parallel with one another;

59. Only the Single PPU read was found in the Preliminary Guidance to obviate claim 28. As described below in connection with claim 29[a], in the Single PPU read, the alleged engines are not in parallel with one another.

b. wherein each of the plurality of tasks is decomposable into two or more related threads

60. Claim 28 requires that engines receive from a scheduling module a plurality of tasks; those tasks must then be received by the shader core, which must execute a first of those tasks while executing a second of those tasks. Wilt fails to disclose that the alleged shader core executes first and second such tasks wherein each of the plurality of tasks is decomposable into two or more threads.

61. A “thread” is a sequence of instructions. Ex. 1001 at 1:47-51. A “task,” in this claim limitation, and in keeping with the ordinary meaning of the term and its

use in the '381 patent, is decomposable into two or more threads, and thus comprises multiple distinct sequences of instructions. *Id.* Wilt operates at the instruction level, not the task level. That is, in Wilt, the shader core receives and allegedly simultaneously executes at best different instructions or threads, not tasks, at the same time. And there is nothing indicating that any such threads would correspond to different parent tasks.

62. Petitioner has alleged that the application software running on the CPU is the claimed scheduling module that transmits tasks that received by the alleged engines and the alleged shader core. Pet. at 26. But in Wilt, the CPU only issues “commands” and a “command stream” that according to Petitioner are ultimately received by the alleged engines and alleged shader core:

In particular, CPU 102 issues commands that control the operation of PPUs 202. In some embodiments, 65 CPU 102 writes a stream of commands for each PPU 202 to a pushbuffer (not explicitly shown in FIG. 1), which may be located in system memory 104, PP memory 204, or another storage location accessible to both CPU 102 and PPU 202. PPU 202 reads the command stream from the pushbuffer and executes commands asynchronously with operation of CPU 102.

Ex. 1004 (Wilt) at 4:63-5:4 (emphases added). The Preliminary Guidance relies on incorrectly equating “command” with “task.” A POSITA would understand that a command is an instruction, two levels of abstraction down from a task.

63. There is no disclosure of modification of these command streams into tasks before transmitting them to the alleged shader core, and thus, at best, this also only discloses that a shader core can receive and execute (allegedly simultaneously) multiple command streams.

64. But without more, executing a command streams does not indicate executing multiple tasks simultaneously, because each command stream is only one thread, not two or more. This is true in both the Single PPU and Multiple PPU Theories. Even if executing a thread of a task were sufficient to indicate execution of a task, and even if one were to view the operation of the alleged shader core over time, there is no evidence that threads from two different tasks would ever be executed at the same time. As Dr. Pedram has admitted and I have explained, in general, looking just at a thread or multiple threads, it is not knowable without more information (that is lacking in the prior art) which task or tasks they were decomposed from; threads could have been decomposed from, and thus correspond to, the same task, for example. Ex. 2007 at 21:6-27:4.

65. Although Wilt states in some portions that cores 208 execute “tasks,” given that the only thing the alleged engines upstream of the shader core ever receive are at best threads, and there is no disclosure of creation of tasks from those threads, the only reasonable interpretation is that the cores actually execute what the ’381 patent would call threads, not tasks as the ’381 patent and the claim language in

claim 28 describe that term. Ex. 1004 at 4:1-45. That understanding is confirmed by other statements in Wilt in the same portion of the reference clarifying that “[e]ach processing core 208 is capable of executing a large number (e.g., tens or hundreds) of threads concurrently. . . .” *Id.* at 4:2-7 (emphasis added).

66. The Preliminary Guidance points to “cooperative thread arrays” and SIMT thread groups. As to SIMT thread groups, the name alone indicates it is only one instruction (Single Instruction Multiple Thread), meaning that there is only one instruction executing. This name also confirms that “thread” in this context is not a sequence of instructions, as it is a single instruction being executed. This is confirmed by the cited quote that “a ‘SIMT thread group’ or ‘warp’ refers to a group of up to P threads of execution of the same program on different input data.” PG at 23-24. As other parts of Wilt state, a “program” is at the level of a thread; it is not a higher-level concept: “each thread is an instance of a program.” Ex. 1006 at 4:1-7. So when the SIMT thread group executes a program, it is operating at the thread level. The same is true of CTAs: “Each CTA is a group of threads that concurrently execute the same program (referred to herein as a ‘CTA program’) on an input data set to produce an output data set.” Ex. 1006 at 9:2-4. Thus, both SIMT thread groups and CTAs execute the same thread and indeed the same instruction at a given time, on different data. Even if they constituted multiple threads, the fact remains that there is no disclosure whatsoever that such threads were definitively decomposed

from different tasks. Wilt does not state any relationship between a “task” and a CTA or SIMT thread group, including at the cited column 6 lines 14-27.

- c. wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.**

67. In Wilt, a “command” enters the PPU via host interface 206, is sent to front end unit 212, then from there to work distribution unit 210, then according to Petitioner and its expert from there to core interface 303, and then to cores 208. Ex. 1004 at 3:49-4:32; Ex. 2007 at 63:24-67:23. The Preliminary Guidance found that the Single PPU Theory of Wilt discloses this element. Patent Owner has added the requirement that “the plurality of engines are arranged in parallel with one another,” which is lacking here as explained regarding claim 29.

68. As to the Multiple PPU Theory, as Dr. Pedram conceded, any core only receives commands from the alleged combined engine within that same PPU, not from more than one engine. Ex. 2007 at 86:24-88:7.

2. Jiao

69. Petitioner has recited two theories as to what constitute the engines in Jiao. Under the Shader Theory, the vertex shader 304, geometry shader 306, and pixel shader 310 are the plurality of engines. Pet. at 39. Under the Thread Slot Theory, the plurality of engines are those items along with the “thread slots” within

each of the execution units 420 of the EU pool 414 that constitutes the alleged shader core. *Id.* at 40.

a. wherein the plurality of engines are arranged in parallel with one another;

70. Jiao's shaders, the alleged engines, are arranged serially, such that input of one relies on the direct output of the other, as depicted in Fig. 3, described in Jiao at column 6, lines 1-48, and conceded by Dr. Pedram. Ex. 2007 at 93:3-16. They are not in parallel with one another. The addition of the "thread slots" as engines in an alternative theory, does not remedy the flaw. The shaders are serially connected to one another and are also serially connected to the shader core, including the thread slots therein, because according to Petitioner's theory, the alleged tasks flow from the shaders to the thread slots within the alleged shader core.

b. wherein each of the plurality of tasks is decomposable into two or more related threads; and

71. There are two places in the claims where tasks decomposable into two or more related threads must be received: by the engines and by the shader core. There are two levels of hierarchy to a task: (1) a task is composed of multiple threads; (2) a thread is composed of a sequence of instructions.

72. The alleged engines, Jiao's shaders, are only ever alleged to receive a series of "commands" from the CPU. Pet. at 40-41. Commands are instructions, and thus two levels away from being a task. They must be organized into a

“sequence” to be a thread, and there must be multiple related threads comprising a task. Jiao does not state that the CPU’s commands are ever received by shaders, only the GPU generally, but even if they are, they are not the claimed tasks.

73. The Preliminary Guidance held that the “operations” performed by the shaders are commensurate with tasks in the ’381 patent. First, that is irrelevant, because there is no disclosure that those “operations” are *received* by the engines. Second, it is incorrect. Those operations, such as “vertex transformations” and “interpolations and other operations that collectively determine pixel colors,” are at the level of instructions or threads. Tasks are much more involved, as Petitioner and its expert agree, citing as an example “drawing a triangle of given shape and size with white boundaries filled with a solid blue color in the right upper corner of a 1920 x1080 LCD Display.” Pet. at 42. That task will have constituent threads and instructions that more accurately correspond to shader operations. In fact, that is *precisely* why Jiao uses distinct, dedicated shaders, one for vertex operations, one for geometry operations, and one for pixel operations, to carry out the full task.

74. As to the shader core, the evidence in Jiao confirms that it operates only at the thread level as well, and when it says “tasks” it does not mean a task that comprises multiple threads. As quoted in the Preliminary Guidance, the uses of “task” and “thread” in the same sentence prove that “task” is what the ’381 patent calls a thread. PG at 18. And “thread” in Jiao is not what the ’381 patent calls a

thread—Jiao uses the term “thread” to mean a *resource*, something that is available for being assigned work and executing work, for example, discussing assignment of “tasks” to “available threads” for execution and that the EU Pool Control “knows which threads have been assigned tasks and are occupied” and where Jiao states “[w]hen a thread completes its assigned task(s)...”, marking the thread as free, etc. There is simply nothing in Jiao with the required relationship between task and thread: that a task is decomposable into two or more related threads; they are wholly distinct concepts, as threads are computational resources, not sequences of instructions. Further, after assignment, “each thread [*i.e.* resources] fetches *an instruction*.” PG at 15. Thus, as alleged tasks are generated and assigned, the thread (resource) fetches constituent instructions—not constituent threads—thus indicating that the “task” is what the ’381 patent calls a single thread. Jiao never discusses generating, assigning, or fetching anything that is decomposable into multiple threads. Again, this makes perfect sense because, the architecture of Jiao is to divide functionality into three shaders; each generates at best a thread corresponding to vertex, geometry, or pixel operations: none of them generates a full task.

75. Further, the shaders in Jiao do not receive tasks comprising multiple threads because the shaders *are* threads, or portions of threads. Jiao states repeatedly that the shaders are “shader programs”: “Various types of shader programs can be executed or mapped to the computational core 204, including the vertex shader 304,

geometry shader 306, and pixel shader 310.” Ex. 1005 at 6:59-62; *see also id.* at 5:49-51. The ’381 patent also discusses “shader programs,” and equates them with threads or portions of threads: “Each thread may comprise a shader program, such as a geometry shader, a pixel shader, a vertex shader, or the like. Each thread (e.g., shader program) is typically associated with a set of state data.” Ex. 1001 at 1:51-54; *see also id.* at 1:58-66, 2:23-31, 7:42-48, 8:33-41. (Notably, other references such as Wilt also use the term “program” as equivalent to a thread, e.g., Ex. 1006 at 4:1-7, stating “each thread is an instance of a program.”). Jiao repeatedly states that the shaders themselves—and the “operations” they perform, which the Board focuses on—*execute on* execution units 420 in EUP 414. Ex. 1005 at 9:27-31; *id.* at 6:59-62. Thus the shaders are shader *programs* that operate on the shader core, and are themselves what the ’381 patent calls threads. But the claims of the ’381 require the engine to be an item that receives and transmits tasks that comprise threads (shader programs). By contrast, Jiao’s shaders are themselves threads or parts of threads, executing on the alleged shader core. They are thus not the claimed engines.

76. Lastly, even if all of the above is incorrect, there is no actual configuration disclosed in Jiao in which the system will definitively simultaneously execute two different threads, each of which is decomposable or has been decomposed into multiple related threads, because there is no showing of what any hypothetical instruction being executed at a given time by the alleged shader core

corresponds to at the task level. At best there is mere happenstance, which is insufficient to meet the “configured to” claim language.

- c. wherein at least one processing element of the multiple processing elements is configured to receive at least partial tasks from each engine.**

77. In contrast with the '381 patent's discussion of explicit configurations that require that a given processing element receives and executes tasks or partial tasks from each engine, Jiao is silent on the matter. There is no actual configuration disclosed in Jiao as to what a given processing element or plurality of processing elements does or receives with respect to the shaders, the alleged engines. If anything, Jiao indicates a dedication of certain execution units to alleged “tasks” coming from particular shaders: “[a]s various shader tasks are generated by the vertex shader 304, geometry shader 306, and pixel shader 310, they are delivered to the respective execution units 420 to be carried out.” Ex. 1005 at 9:35–38. That is, a task from a given shader is sent to the respective execution unit for that shader. At a minimum there is certainly no specific configuration as claimed.

B. Claim 29

1. Wilt

- a. wherein the plurality of engines are arranged in parallel with one another**

78. In the Single PPU Theory, the plurality of engines Petitioner has pointed to are arranged in series with one another, not in parallel, as Dr. Pedram

conceded. Ex. 2007 at 67:24-68:2. Front end 212 sends threads to work distribution unit 210, which Dr. Pedram alleges sends them to core interface 303, which sends them to core 208 for execution.

b. wherein each engine is configured to receive at least one of the plurality of tasks via a distinct command buffer for each engine

79. Under either Wilt theory (the Single PPU Theory or the Multiple PPU Theory), the alleged engines in Wilt do not each have a distinct command buffer for each engine, as Dr. Pedram conceded. In the Single PPU Theory, front end 212, work distribution unit 210, and core interface 303 are not described as each having a distinct command buffer, much less one from which the alleged engines receive the alleged tasks. In the Multiple PPU Theory, the Preliminary Guidance held that the host interface 206 is a command buffer for the amalgamated “engine” in each PPU. A POSITA would not have understood the host interface 206 to satisfy the requirements of the claimed buffer. A POSITA would have understood the host interface 206 to be simply an interface to memory bridge 105, which then connects to global system memory 104. Ex. 1006 at 3:56-60, 4:24-30; 7:15-19. A POSITA would have understood the host interface 206 to simply pass through commands from the memory bridge 105 and ultimately from global system memory 104 to the PPU. A POSITA would not have understood a buffer to be inherent or obvious in this transmission process.

80. The claimed command buffer receives and stores tasks, and buffers them for receipt by the engines. Ex. 1001 at 10:22-39. In Wilt, by contrast, there is a global system memory 104, and host interface 206 passes commands from this memory to the alleged engines. A POSITA would not have understood this interface to perform any buffering or store any tasks, nor would have found any motivation for it to do so, given its purpose and given the multi-processor bus access to a shared system memory 104.

- c. wherein the scheduling module transmits tasks to command buffers for the engines and determines which task to transmit to which engine's command buffer based on a priority level associated with the task,**

81. Wilt contains no discussion of priority, much less does it disclose task priority or the remainder of the requirements of this limitation. Indeed, there is no disclosure whatever of how the alleged scheduling module transmits tasks.

- d. wherein each of the plurality of tasks is decomposable into two or more related threads; and**

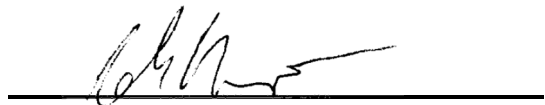
82. As discussed above with respect to claim 28, Wilt's alleged engines and the alleged shader core only receive and execute at best individual threads, and so the shader core does not simultaneously execute different tasks.

- e. wherein each engine is configured to load state data associated with at least one of the plurality of tasks**

83. Only core interface 303 is alleged to load state data. Front end unit 212 and work distribution unit 210 do not load state data, as Dr. Pedram admitted. Ex. 2007 at 73:14-74:3.

May 22, 2024

Respectfully Submitted,



Dr. John Hart