Trials@uspto.gov
Tel: 571-272-7822

Paper No. 7
Date: October 25, 2023

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,

v.

ATI TECHOLOGIES ULC,
Patent Owner.

IPR2023-00687
Patent 8,854,381 B2

Before JAMES P. CALVE, BRIAN J. McNAMARA, and KEVIN W. CHERRY, *Administrative Patent Judges.*

CALVE, *Administrative Patent Judge*.

DECISION
Granting Institution of *Inter Partes* Review
35 U.S.C. § 314

I. BACKGROUND

Realtek Semiconductor Corp. ("Petitioner") filed a petition seeking an *inter partes* review of claims 15–20 (the "challenged claims") of U.S. Patent No. 8,854,381 ("the '381 patent") (Ex. 1001). Paper 1 ("Pet."), 1, 3. ATI Technologies ULC ("Patent Owner") timely filed a Preliminary Response. Paper 6 ("Prelim. Resp.").

We have authority to determine whether to institute an *inter partes* review. *See* 35 U.S.C. § 314 (2018); 37 C.F.R. § 42.4(a) (2022) ("The Board institutes the trial on behalf of the Director."). An *inter partes* review may not be instituted "unless . . . the information presented in the petition . . . shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition." 35 U.S.C. § 314(a). Upon consideration of the evidence and arguments in the record, we determine that the information presented shows a reasonable likelihood that Petitioner would prevail with respect to at least one of the challenged claims. We therefore grant institution of an *inter partes* review.

A. Related Proceedings

The parties identify the following proceedings involving the '381 patent: *Advanced Micro Devices, Inc. v. TCL Indus. Holdings Co., LTD*, No. 2:22-cv-00134 (E.D. Tex.) and *Certain Graphics Systems, Components Thereof, and Digital Televisions Containing the Same*, Inv. No. 337-TA-1318 (U.S. Int'l Trade Comm.) ("ITC proceeding"). Pet. 1–2; Paper 4, 1.

B. Real Parties in Interest

Petitioner identifies itself as the real party in interest. Pet. 1. Patent Owner identifies Advanced Micro Devices, Inc. ("AMD") as the real party in interest because it owns all right, title, and interest in the '381 patent and asserts that ATI is a wholly-owned indirect subsidiary of AMD. Paper 4, 1.

C. The '381 Patent (Ex. 1001)

The '381 patent address the problem of context switching that occurs when a graphics-processing unit (GPU) changes a task that it is processing and replaces the stored state data (e.g., texture handles, shader constants) for the current task with state data for a new task. Ex. 1001, 2:8–37.

A context switch requires state data associated with the first task to be swapped into back-up storage units of the GPU so new state data associated with a more important task can be retrieved and placed into the data-storage units of the shader core. Ex. 1001, 2:23–28. Then, the shader core executes the more important task using the new state data in the data storage units. *Id.* at 2:28–31. After the more-important task finishes executing, state data for the more important task are flushed from data storage units, and the state data for the first task are swapped back into data storage units of the shader core. Then, the shader core resumes executing the first task. *Id.* at 2:31–37.

Context switching takes substantial time and memory to store context that is switched. Ex. 1001, 2:38–45. The time required to perform a context switch limits GPU performance, and the additional local memory requires a larger GPU. *Id.* The time needed for a context switch (e.g., hundreds of clock cycles) makes the effective time of executing a high priority task relatively long even through the actual time for executing the high priority task may be relatively short (e.g., tens of clock cycles). *Id.* at 2:51–56.

To address this issue, the '381 patent embodies plural virtual engines 202 on a shader core 214 of a GPU as shown in Figure 2, reproduced below.

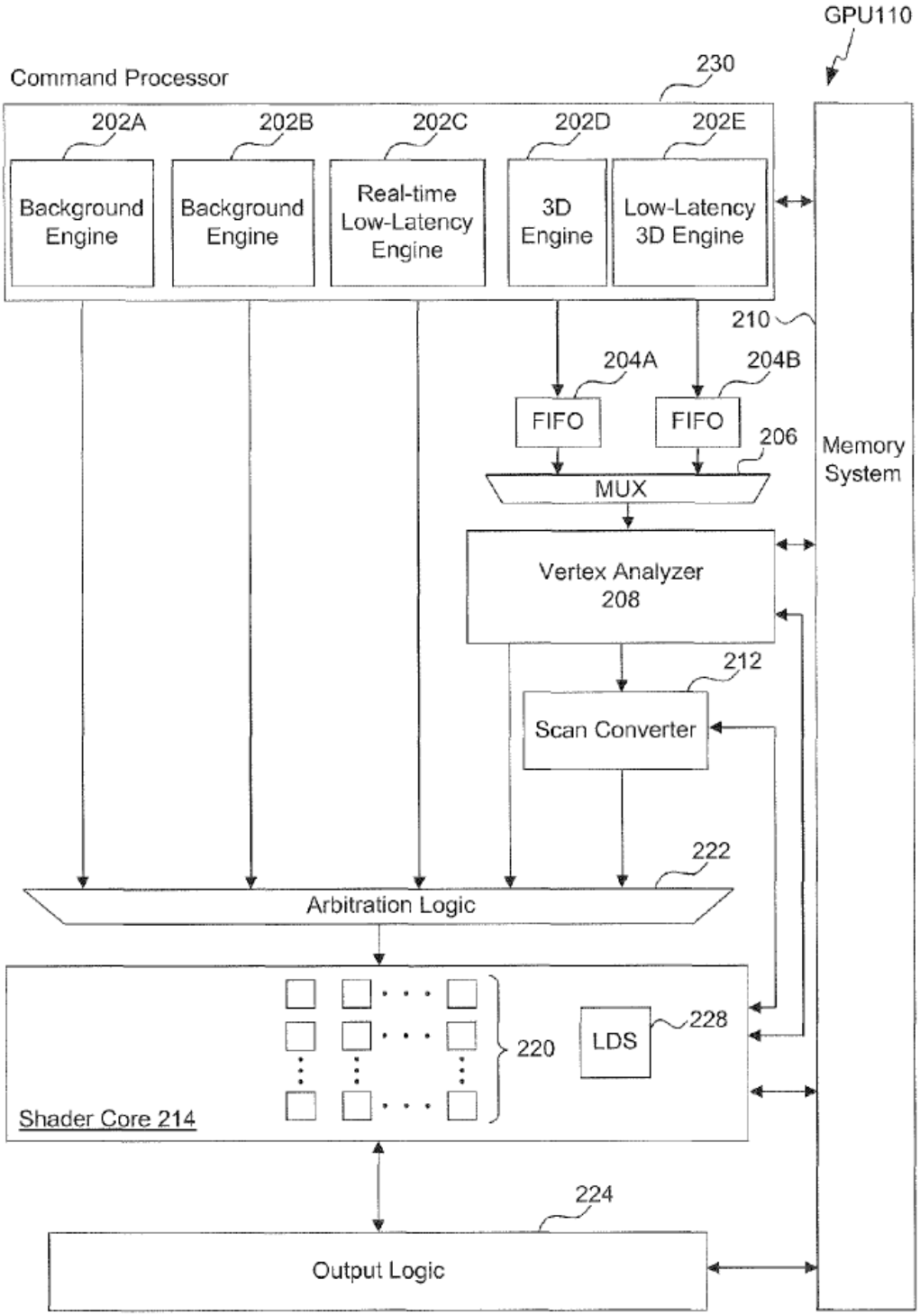

FIG. 2

Figure 2 of the '381 patent depicts GPU 110 comprising a plurality of virtual engines embodied on a shader core. Ex. 1001, 6:42–43. Each virtual engine 202 is configured to retrieve tasks provided by an OS scheduler to command buffers 420 of the GPU and issue each task to a single shader core substantially in parallel. *Id.* at 2:8–19, 5:4–13, 6:65–7:4, 10:14–41, Fig. 4.

The virtual engines enable GPU 110 to execute different streams of processing tasks received from the OS scheduler simultaneously without a context switch. GPU 110 executes tasks that correspond to the different contexts in a single shader core by sharing resources of shader core 214 among tasks. Ex. 1001, 6:50–54. Shader core 214 includes local data shares (LDS) 228 to store the context state data for different tasks. *Id.* at 8:23–32.

Background engines 202A, 202B process low-priority tasks and take over resources of GPU 110 only when no other virtual engines are using the resources of GPU 110. Ex. 1001, 7:6–12. Real-time, low-latency engine 202C has priority access to the resources of GPU 110 to process high-priority tasks. *Id.* at 7:12–14. Primary 3D engine 202D processes standard graphics-processing tasks. Low-latency 3D engine 202E processes high-priority graphics-processing tasks and has priority access to the graphics-processing resources of GPU 110. *Id.* at 7:14–19.

Shader core 214 resources are partitioned in space and/or time so it can process tasks that are received from virtual engines 202 substantially simultaneously. Spatial partitioning issues a first task (e.g., a low latency task) from a first virtual engine to a first subset of processing elements 220 of shader core 214 and a second task (e.g., a standard graphics task) from a second virtual engine to a second subset of processing elements 220 of shader core 214. Ex. 1001, 5:14–22, 8:8–17.

For temporal partitioning, two tasks share a percentage of time of processing elements 220 of shader core 214. Ex. 1001, 5:22–28, 8:18–22.

Tasks from command processor 230 are provided to input logic that schedules tasks for execution in shader core 214 based on the availability of resources in shader core 214 and the priority of tasks. Ex. 1001, 7:20–29. Tasks from primary 3D engine 202D and low-latency 3D engine 202E are sent to first-in, first-out (FIFO) buffers 204A, 204B and then to multiplexer 206, which provides tasks to vertex analyzer 208. *Id.* at 7:34–41.

Vertex analyzer 208 identifies the shader programs that are associated with a graphics-processing or general-compute task, schedules the launch of shader programs in shader core 214, and generates pointers for vertices. It sends the graphics-processing tasks to scan converter 212 and the general-compute tasks to arbitration logic 222. Ex. 1001, 7:55–59. Arbitration logic 222 includes multiplexers to provide tasks from different virtual engines of command processor 230 to shader core 214. *Id.* at 7:60–62.

Shader core 214 includes processing elements 220 that execute tasks provided to GPU 110 and that are arranged as SIMD devices so shader core 214 can execute data-parallel processing tasks substantially simultaneously. Ex. 1001, 7:66–8:22. Local data shares 228 store state data associated with each task executed by shader core 214. *Id.* at 8:23–32.

Figure 4, reproduced below, shows how tasks 404A–D received from applications 402A, 402B running on CPU 102 are executed on GPU 110 by shader core 214. Ex. 1001, 10:1–19. OS scheduler 310 provides tasks 404 to buffers 420A–C serviced by virtual engines 432A–N that execute tasks on shader core 214 according to the priority and availability of resources using scheduling module 434. *Id.* at 10:33–55.

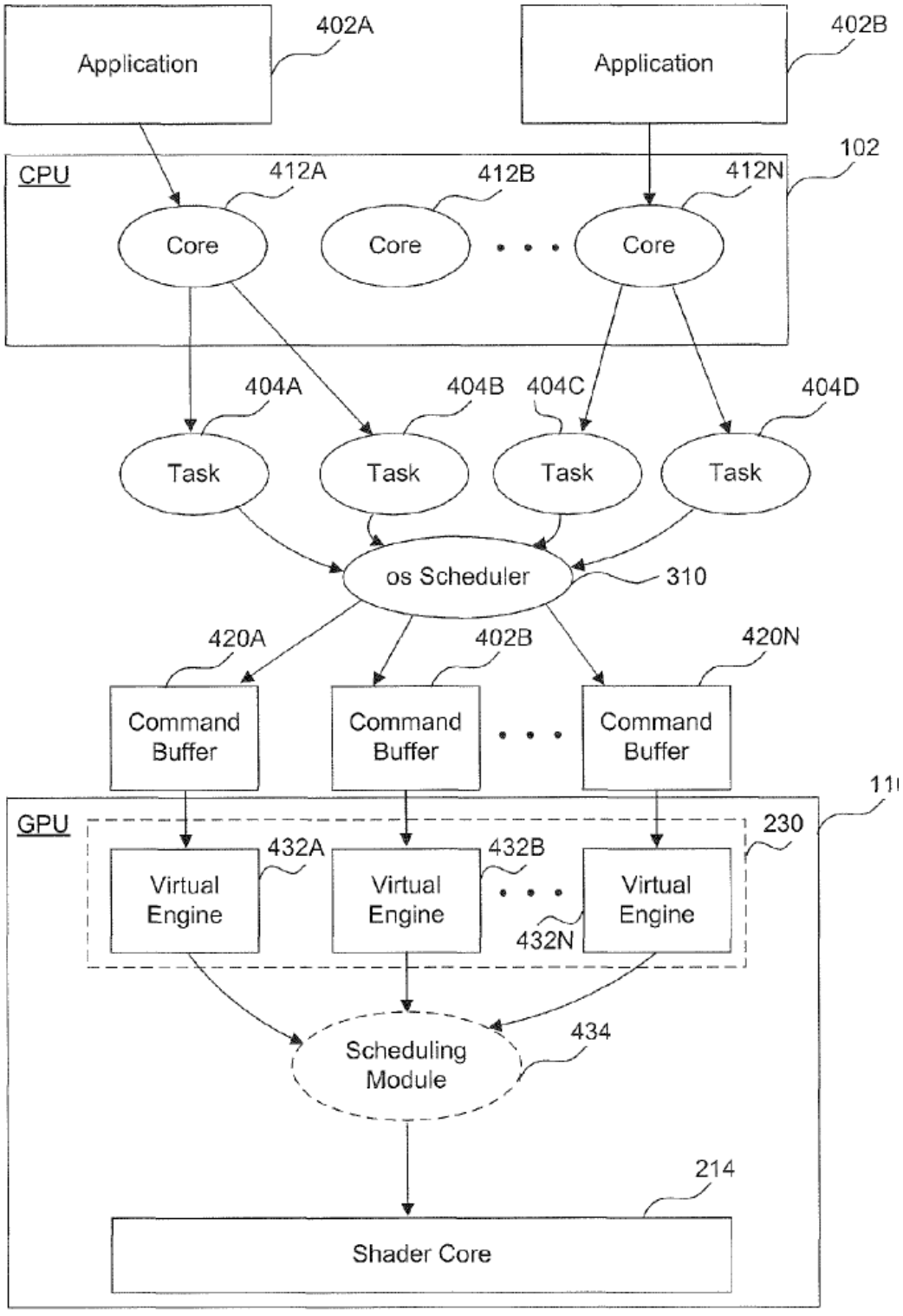

Figure 4 illustrates how different streams of processing tasks are executed by virtual engines 432 in shader core 214. Ex. 1001, 10:1–55.

D. Challenged Claims

Claim 15 is independent with claims 16–20 depending therefrom and is reproduced below with Petitioner's annotations to identify each limitation.

15[preamble] A computer system comprising:
15[a] a memory;
15[b] a first processing unit; and
15[c] a bus coupled to the memory and the first processing unit,
15[d] 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
15[e] 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.

E. Asserted Grounds of Unpatentability

Petitioner asserts the following grounds of unpatentability (Pet. 21):

Ground	Challenged Claims	35 U.S.C. §	Reference(s)
1	15–20	103(a)[1]	Wilt[2]
2	15–20	103(a)	Jiao[3]
3	15–20	103(a)	Stuttard[4]

[1] The Leahy-Smith America Invents Act ("AIA") revised 35 U.S.C. § 103 effective on March 16, 2013. Because the '381 patent has an effective filing date before March 16, 2013, we use the pre-AIA version of 35 U.S.C. § 103.

[2] US 8,151,095 B1, issued Apr. 3, 2012 (Ex. 1004, "Wilt").

[3] US 8,319,774 B2, issued Nov. 27, 2012 (Ex.1005, "Jiao").

[4] WO 00/62182, published Oct. 19, 2000 (Ex. 1006, "Stuttard").

II. DISCUSSION

A. Level of Ordinary Skill in the Art

Petitioner's declarant, Dr. Pedram, applies a definition of the level of ordinary skill from the ITC proceeding. Patent Owner does not contest it.

> [A] [skilled artisan] would have a bachelor degree in electrical engineering, computer engineering, computer science, or a related field, and at least two years of experience in 'parallel computing or graphics processing including developing, designing, or programming software or hardware for graphics processing or parallel processing units, graphics accelerators[,] or other graphics processing systems,' with additional education substitution for experience and vice versa.

Ex. 1003 ¶ 38 (citing Ex. 1008, 9). We adopt this definition because it is consistent with the '381 patent and the asserted prior art.

B. Claim Interpretation

We interpret claims "using the same claim construction standard that would be used to construe the claim[s] in a civil action under 35 U.S.C. 282(b)" "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). We construe the claims only to the extent necessary to determine whether to institute *inter partes* review. *See Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017).

1. "associated with"

A "scheduling module associated with a second processing unit" in claim 15 is not in a first processor. This meaning is consistent with the '381 patent prosecution history where Patent Owner argued a prior art scheduling module in the first processing unit with the engines was not "associated with a second processing unit." *See* Ex. 1002, 183; *see also* Ex. 2003, 158–161.

2. Remaining Terms

Having considered the record, we determine that no other express claim construction is necessary for this decision. We apply the plain and ordinary meaning of the remaining claim terms. Any final written decision entered in this case that differs from the discussion of claim scope provided in our analysis below may include a final claim construction based on the full trial record.

C. Principles of Law

A patent claim is unpatentable under 35 U.S.C. § 103 if differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. 35 U.S.C. § 103; *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). "The combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *KSR*, 550 U.S. at 416. Similarly, "if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.* at 417.

The question of obviousness is resolved based on underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) when in evidence, objective evidence of non-obviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). Neither party presents any objective evidence of non-obviousness at this stage of the proceeding.

D. Ground 1: Alleged Obviousness Over Wilt

Petitioner asserts that claims 15–20 are unpatentable under 35 U.S.C. § 103(a) over Wilt. Pet. 3, 21–35.

1. Wilt (Ex. 1004)

Wilt discloses computing system 100 with its CPU 102 coupled to system memory 104 and parallel processing subsystem 112 via memory bridge 105 as shown in Figure 1, reproduced below. Ex. 1004, 2:59–3:9.

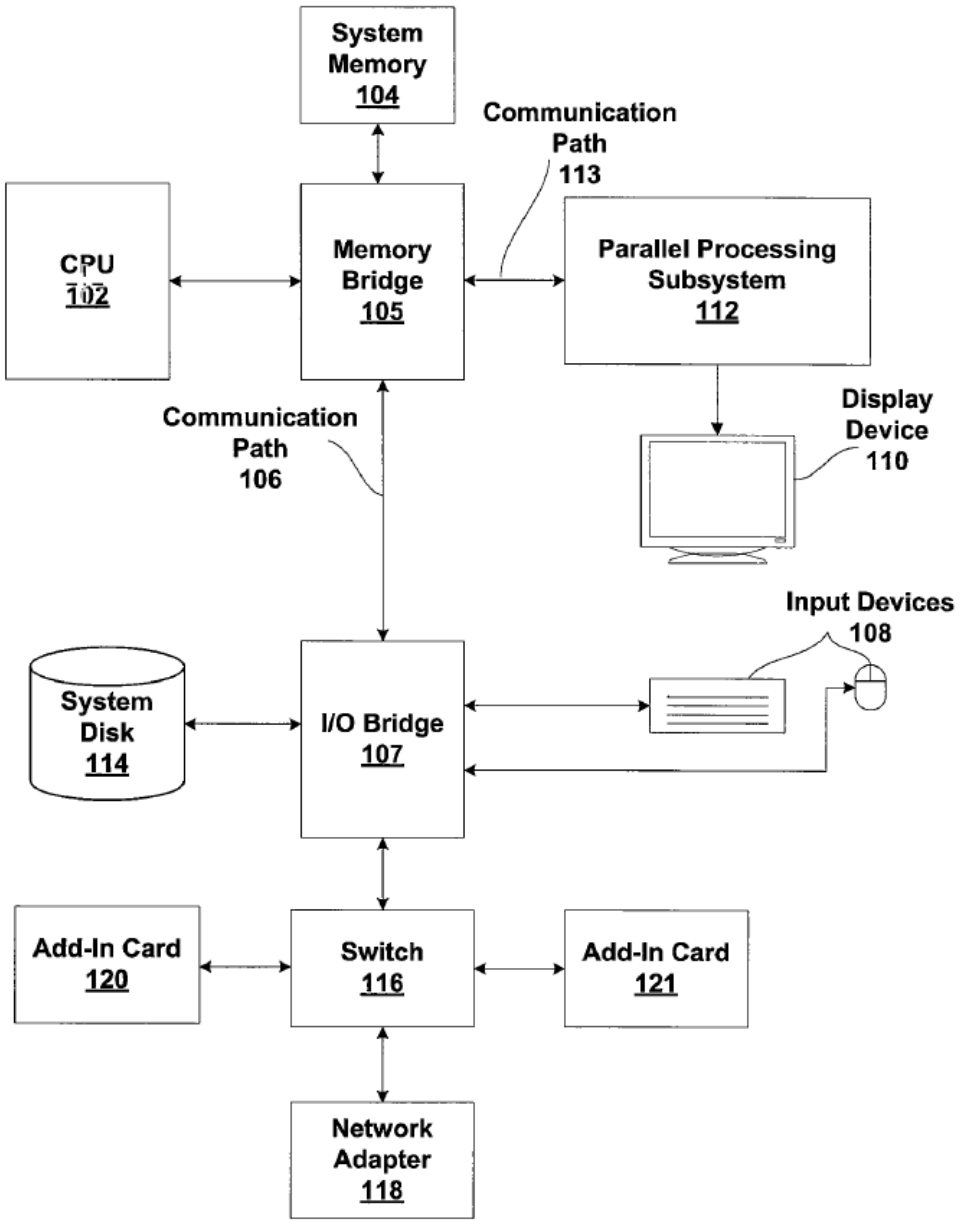

Figure 1 of Wilt shows system 100 with parallel processing subsystem 112 and CPU 102. Ex. 1004, 1:7–9, 2:59–3:5. Parallel processing system 112 may be a graphics subsystem that comprises parallel processing units (PPUs) 202, shown in Figure 2, reproduced below. *Id.* at 3:6–9.

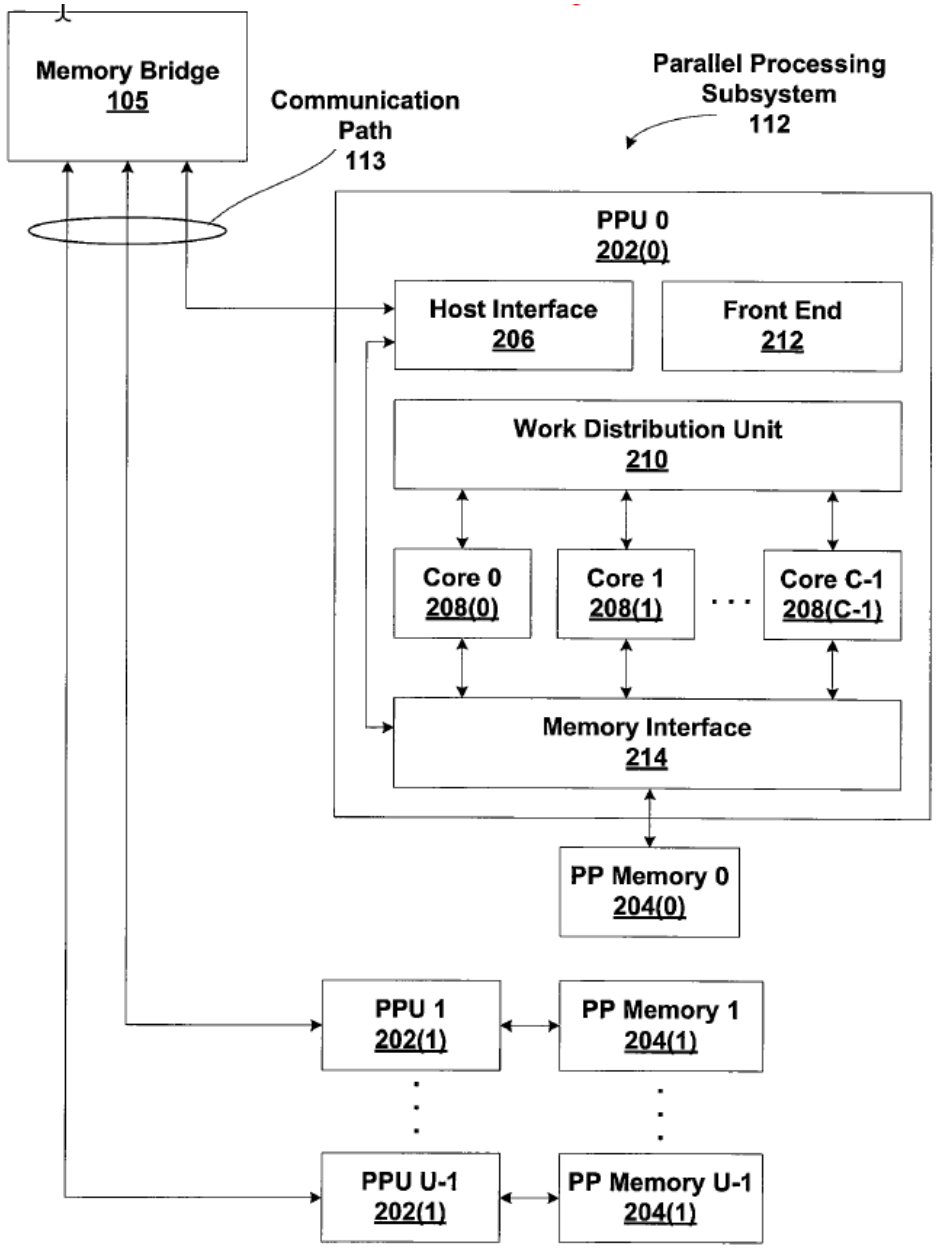

Figure 2

Figure 2 above depicts components of PPU 202. Ex. 1004, 3:36–47.

CPU 102 issues commands that control the operation of PPUs 202 and may write a stream of commands for each PPU 202 to a pushbuffer located in system memory 104 or PP memory 204. Ex. 1004, 4:61–67. PPU 202 reads the command stream from the pushbuffer and executes the commands asynchronously with the operation of CPU 102. *Id.* at 5:2–4.

Each PPU 202 includes host interface 206 that communicates with the rest of system 100 via communication path 113. Host interface 206 receives all incoming packets (and signals) from communication path 113 and directs them to the appropriate components of PPU 202. Ex. 1004, 3:49–60. Host interface 206 receives commands related to processing tasks and directs the commands to front end unit 212. *Id.* at 3:60–64.

Work distribution unit 210 receives commands defining processing tasks from front end unit 212 and distributes the processing tasks to cores 208 to execute. Ex. 1004, 4:7–10. Work distribution unit 210 implements various algorithms to distribute tasks to cores 208 such as assigning a task to a core that asserts a "ready" signal to indicate it has resources to accept a new processing task, and holding a task if no core is ready. *Id.* at 4:10–19.

Core interface 303 controls operation of cores 208 by receiving data to be processed with state parameters and *commands* that define how data is to be processed (e.g., what program is to be executed) *from work distribution unit 210.* Ex. 1004, 8:14–19. Core interface 303 loads data to be processed into shared memory 306 and state parameters into parameter memory 308. *Id.* at 8:19–21. Core interface 303 initializes each new thread or single-instruction multiple-thread (SIMT) thread group in instruction unit 312 and signals instruction unit 312 to begin executing the threads. *Id.* at 8:22–24. Figure 3 of Wilt is reproduced below to illustrate this configuration.

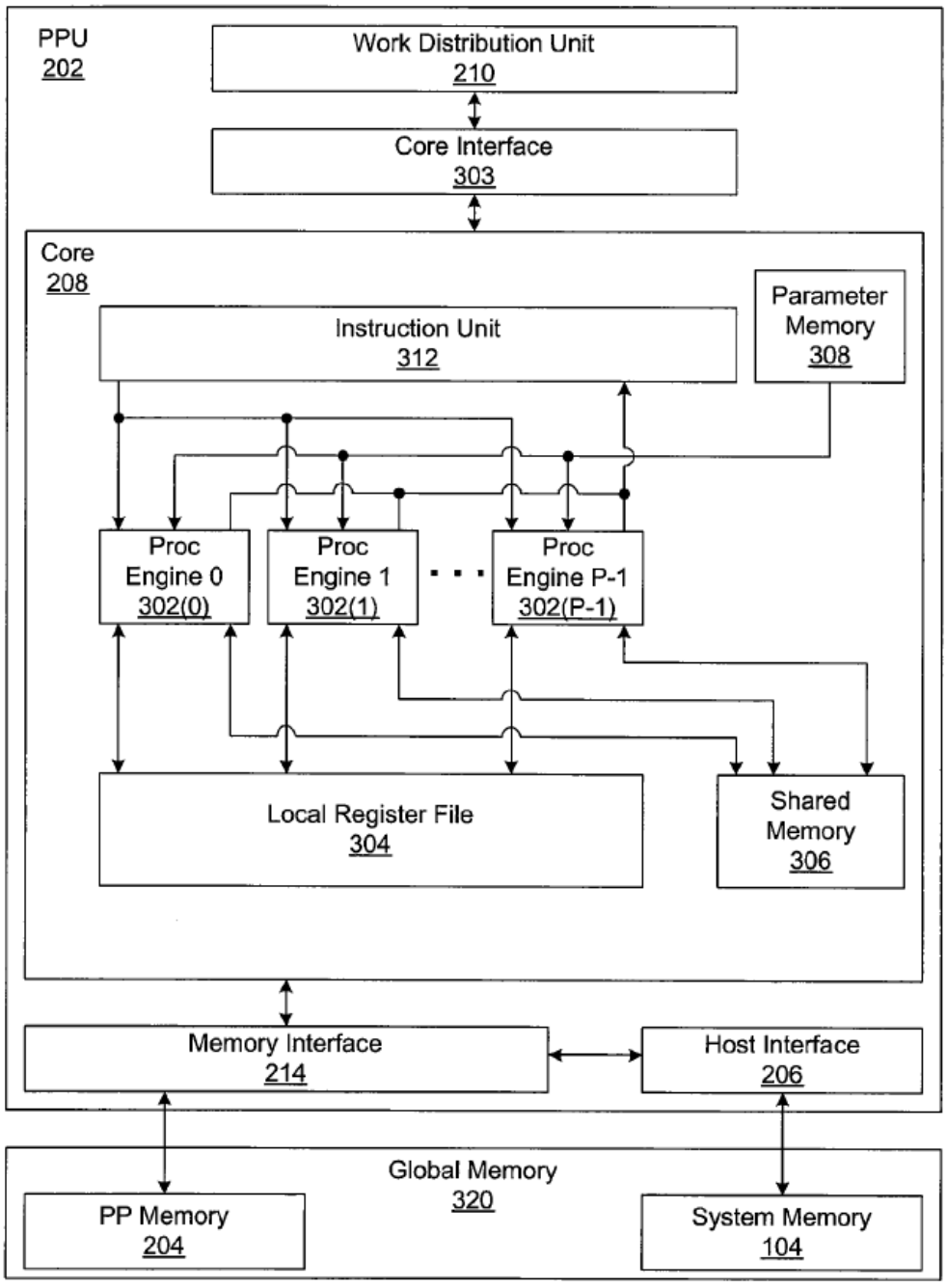

Figure 3

Figure 3 of Wilt shows PPU 202 with work distribution unit 210, core interface 303, instruction unit 312, core 208, and processing engines 302 that execute threads in parallel. PPU 202 includes multiple cores 208 that can operate independently on different tasks. Ex. 1004, 5:59–6:6, 8:14–44.

2. Independent Claim 15

a. 15[preamble]: "A computer system comprising:"

Petitioner asserts that Wilt discloses computing system 100 with CPU 102 and system memory 104 communicating over a bus path and memory bridge 105. Parallel processing subsystem 112 is coupled to memory bridge 105 by a bus or communication path 113 and may be a graphics subsystem that delivers pixels to display device 110. Pet. 21–22 (citing Ex. 1004, 2:59–3:9, 3:24–35, Fig. 1; Ex. 1003 ¶ 64). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

b. 15[a]: "a memory;"

Petitioner contends that Wilt's computing system comprises system memory 104. Pet. 23 (citing Ex. 1004, 2:59–3:3, Fig. 1; Ex. 1003 ¶ 65). This contention, which Patent Owner does not contest at this stage, is supported by record evidence.

c. 15[b]: "a first processing unit;"

Petitioner asserts that Wilt's parallel processing subsystem 112 is "a first processing unit" and it comprises plural PPUs 202 with each coupled to parallel processing memory 204. Petitioner asserts that each PPU 202 has a number of cores 208 that receive processing tasks to be executed via work distribution unit 210 and execute a large number of threads simultaneously. Petitioner asserts that core interface 303 controls operation of each core 208, receives data to be processed, and loads data into shared memory 306 and parameters into parameter memory 308. Pet. 23–25 (citing Ex. 1004, 3:36–48, 4:1–23, 4:33–45, 8:14–29, Figs. 1, 2; Ex. 1003 ¶¶ 66–69). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

d. 15[c]: "a bus coupled to the memory and the first processing unit,"

Petitioner contends that Wilt discloses "a bus coupled to the memory and the first processing unit" as a bus or other communication path 113 that is coupled to parallel processing subsystem 112 (*and PPUs 202*) and system memory 104 via memory bridge 105. Pet. 26 (citing Ex. 1004, 2:59–3:9, 3:49–67, Figs. 1, 2; Ex. 1003 ¶ 70). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

e. 15[d]: "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"

Petitioner asserts that Wilt discloses a "plurality of engines" as work distribution unit 210, front-end unit 212, and core interface unit 303 in each PPU, and these engines receive and process tasks from a scheduling module that is software programs running on the host CPU, e.g., a "driver program." Pet. 26 (citing Ex. 1004, 3:36–67, 4:1–23, 4:46–5:3, 8:14–29, 9:1–17, 9:63–10:31, Figs. 1–3; Ex. 1003 ¶ 71). *Alternatively*, Petitioner asserts that work distribution unit 210, front-end unit 212, and core interface unit 303 are one claimed "engine," and the engines of plural PPUs 202 constitute the claimed "plurality of engines." *Id.* at 27 (citing Ex. 1003 ¶ 73). Petitioner asserts that the "scheduling module" is application software running on CPU 102 such as a "driver program executing on CPU 102," and core interface 303 loads "state data" as state parameters for each command thread into parameter memory 308 of core 208. *Id.* at 26–27 (citing Ex. 1004, 1:53–57, 3:3–7, 8:14–29, 10:21–36; Ex. 1003 ¶¶ 72–74).

f. Do Wilt's engines 210, 212, 303 receive tasks?

Patent Owner argues that Wilt's engines (i.e., work distribution unit 210, front-end unit 212, and core interface unit 303) do not receive tasks from a scheduling module such as "software programs running on the host CPU" because only Host Interface 206 receives tasks from such programs. The alleged engines 210, 212, and 303 do not. Prelim. Resp. 36–38.

The plain meaning may be (1) engines are configured to receive tasks directly/only from a scheduling module as Patent Owner asserts (*see* Prelim. Resp. 36–38) or (2) engines are configured to receive tasks that a scheduling module issues even if the tasks are received via intermediary elements.

We reject meaning 1 because the '381 patent describes engines being *configured to retrieve tasks from command buffers* that are filled by an OS scheduler 310. Ex. 1001, 5:4–13, 10:14–39, Fig. 4. Each virtual engine is "configured to service one of command buffers 420A–N." *Id.* at 10:33–34. To pass tasks to the GPU, an OS scheduler places the tasks in a command buffer. *Id.* at 2:8–9. Command processor 230 and virtual engines 202A–E "receive tasks . . . from one or more command buffers filled by the OS scheduler," and the engines process different tasks. *Id.* at 6:65–7:4. Engines thus *receive tasks from OS scheduler* by retrieving each task from a buffer.

We adopt meaning 2 as more consistent with the '381 specification, which describes the virtual engines as processing different types of tasks that they retrieve from command buffers filled by an OS schedule. Engines are *configured to retrieve processing tasks from command buffers filled by an OS scheduler* and process the tasks. Ex. 1001, 5:4–13, 6:65–7:17, 10:14–41. "Each virtual engine is configured to receive data-parallel tasks . . . and independently execute these tasks on the single shader core." *Id.* at 4:48–51.

Each PPU 202 of parallel processing system 112 is a first processing unit. It has plural engines 210, 212, 303 and shader core 208. Ex. 1004, 3:36–4:45, 5:59–6:27, Figs. 1–3. Host interface 206 of each PPU 202 reads *commands for processing tasks* received from a driver program on CPU 102 via a pushbuffer. *Id.* at 3:49–67, 4:61–5:51. Each front end unit 212 engine receives task commands from host interface 206. *Id.* at 3:56–61. Work distribution unit 210 engine receives commands defining tasks from front end unit 212 and distributes them to cores 208. *Id.* at 4:7–11. Core interface 303 engine receives data to process, parameters, and commands defining how to process the data (what program to execute) from work distribution unit 210, loads data into memory 306 and state parameters into memory 308 of core 208, and launches threads. *Id.* at 8:14–21, 10:21–36; Pet. 26–27.

> *g. 15[e]: "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."*

Petitioner asserts that "[e]ach parallel processing unit (PPU) 202 is the claimed 'shader core configured to receive the plurality of tasks' from its work distribution unit and the front end unit and to execute concurrent tasks (using its processing cores 208)" that are single-instruction multiple-thread (SIMT) parallel (multithreaded) cores 208 that each can execute hundreds of threads inside each PPU. Pet. 28 (citing Ex. 1004, 3:36–48, 4:1–5:3, 5:41–6:52, 6:66–7:57, Figs. 1–3; Ex. 1003 ¶¶ 75, 76). Petitioner asserts the shader core can be plural PPUs 202 that receive tasks from work distribution units and front end units and execute concurrent tasks using processing cores 208 using state parameters associated with the tasks. *Id.* (citing Ex. 1003, ¶ 77).

h. Does Wilt's shader core receive tasks from engines?

Patent Owner argues that claim 15 requires "a shader core configured to ***receive*** the plurality of tasks ***from*** at least one of the plurality of engines," and engines cannot be inside the shader core because then the shader core would not receive the tasks from the engines. Prelim. Resp. 31–32. Patent Owner argues that Petitioner's engines are *within* the alleged "shader core" and thus cannot be the claimed "engines" under either theory. *Id.* at 33–36.

Petitioner's assertion that PPU 202 has a work distribution unit and front end unit that execute tasks on processing cores 208 treats PPU 202 as a "first processing unit" because each PPU 202 has work distribution unit 210 and front end unit 212 engines as discussed for element 15[d] and processing cores 208 receive and execute tasks from those engines. Pet. 26–28. PPU 202 does not receive or execute tasks. Its engines receive tasks that its cores execute as Petitioner asserts. *Id.* It is unclear how subsystem 112 is used when each PPU 202 of subsystem 112 discloses elements 15[d] and 15[e].

Wilt's work distribution unit 210 receives tasks from front end unit 212 and assigns the tasks to each core 208. Ex. 1004, 4:1–23. Petitioner alternatively asserts that units 210, 212, 303 of each PPU 202 are one engine that assigns the tasks to cores 208 on plural PPUs 202. Pet. 27–28.

i. Do Wilt's engines load state data?

Patent Owner argues that each individual engine must be configured to load state data for each of the plurality of tasks, and the Petition only alleges that core interface 303 loads the state data. Prelim. Resp. 40–41.

Claim 15 does not require *each* of the plurality of engines to receive a plurality of tasks or each of the engines "to load state date associated with each of the plurality of tasks." Ex. 1001, 14:5–9.

A plurality of engines receives a plurality of tasks and loads the state data for *each* task. State data is loaded for *each* task, but any of the plurality of engines can do so. Ex. 1001, 14:5–9, 3:1–11. The '381 patent indicates a plurality of tasks are received from an OS scheduler and "a set of state data associated with each of the plurality of tasks is loaded." *Id.* at 3:30–34.

The '381 patent describes engines 202A–E processing different types of tasks but does not describe how the engines load state data for any tasks. *See* Ex. 1001, 7:1–21. The '381 patent describes state data associated with each task being stored in local data shares (LDS) 228 but does not describe how the state data is loaded into LDS 228 by an engine(s). *Id.* at 8:23–32.

Wilt's core interface 303 of each PPU 202 loads the state parameters into memory 308 of each core 208. Ex. 1004, 10:21–36. Patent Owner does not contest this contention of Petitioner. *See* Pet. 27; Prelim. Resp. 40.

j. Does Wilt's shader core execute first and second tasks?

Patent Owner argues that Wilt's processing core 208 only executes a single parallel program on different data across plural parallel processing engines 302 in core 208. Prelim. Resp. 41. Patent Owner asserts that "[a]t best, Wilt discloses multiple *threads* executing simultaneously." *Id.* at 42.

In Wilt, the shader core of PPU 202 has multiple cores 208 that each can execute a *task* for an *application*, and work distribution unit 210 assigns new tasks to each core 208. Ex. 1004, 4:1–40, 6:2–52, 7:43–57. Cores have the same design so the tasks execute the same on the different cores. *Id.* at 8:14–44. Alternatively, Petitioner asserts that the shader core can be plural PPUs 202, each of which processes the tasks independently. Pet. 28. Patent Owner does not contest this alternative contention. Prelim. Resp. 41–43.

3. Claims 16–20

Petitioner asserts that Wilt discloses that each core 208 in PPU 202 has plural parallel processing engines 302 each of which executes instruction streams for the same SIMT thread groups, "different cores 208 can execute different instruction steams in different SIMT thread groups [so] cores 208 can simultaneously execute different tasks (each task comprising its own SIMT thread group) based on their associated state data (which is stored in the parameter memory 308 of each processing core 208)," and cores 208 can execute multiple tasks simultaneously based on their associated state data so the cores inside each PPU 202 can be grouped to execute first and second tasks based on associated state data as in claim 16. Pet. 29–32.

Petitioner asserts that Wilt discloses that PPUs 202 include rendering pipelines for their graphics processors configured for various tasks using a pipelined architecture that would render obvious the first and second queues configured to receive the first and second tasks in claim 17. *Id.* at 32–33.

Petitioner asserts that Wilt discloses different latencies for data access and processing in memory for threads and a skilled artisan would understand that different implementations result in different access latencies for tasks as in claim 18. *Id.* at 34–35. Petitioner asserts that Wilt's parallel processing subsystem of PPUs processes graphics and general tasks in claim 19, and the shader core receives tasks based on priority as in claim 20 using a context stack. *Id.* at 35. These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

Accordingly, on this record, we find that Petitioner has established a reasonable likelihood that claims 15–20 are unpatentable over Wilt.

E. Ground 2: Alleged Obviousness Over Jiao

Petitioner asserts that claims 15–20 are unpatentable under 35 U.S.C. § 103(a) over Jiao. Pet. 3, 36–52.

1. Jiao (Ex. 1005)

Jiao discloses graphics processing system 100 having GPU 114 that includes graphics processor 202 that comprises plural execution units as depicted in Figure 4, which is reproduced below. Ex. 1005, 2:24–36.

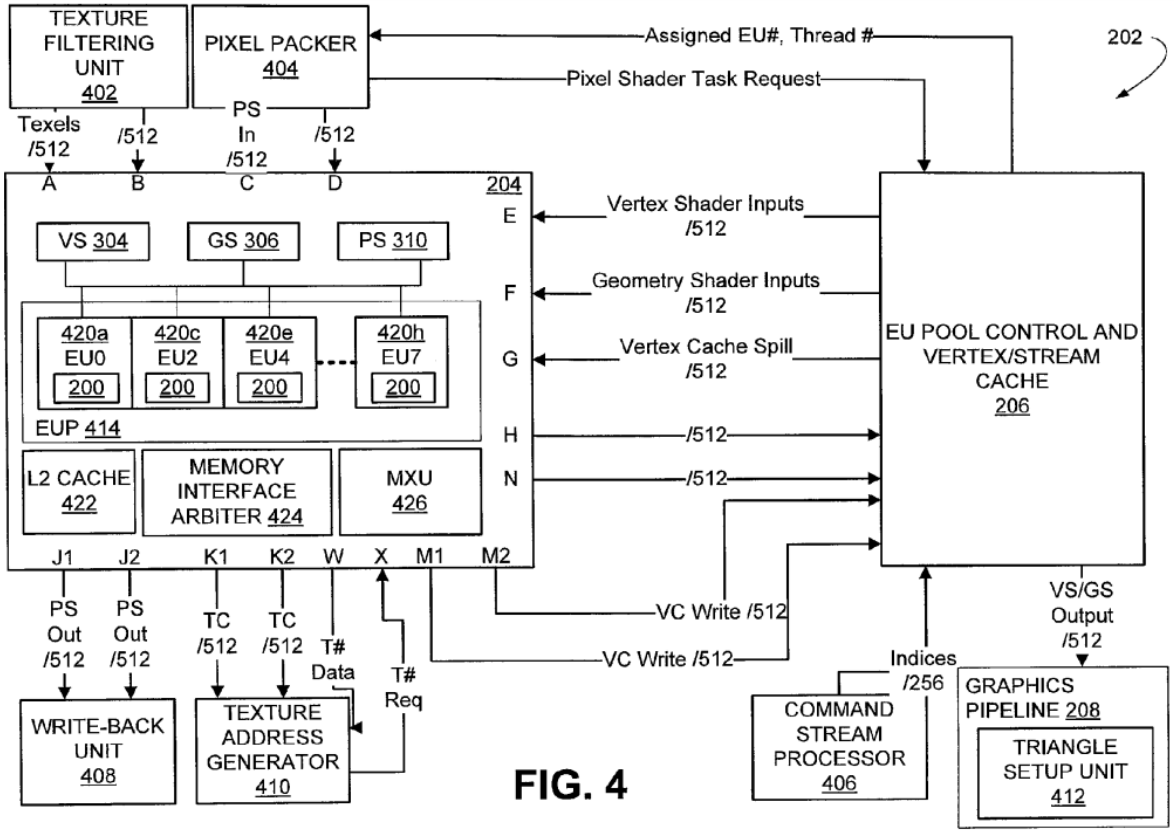

FIG. 4

Figure 4 depicts graphics processor 202 with computational core 204 comprising vertex shader 304, geometry shader 306, and pixel shader 310 that receive and process graphics tasks received from driver software 128 of system memory 124 via EU Pool Control 206 on execution units (EUs) 420 of execution unit pool (EUP) 414. Ex. 1005, 6:49–8:34.

2. Independent Claim 15

a. 15[preamble]: "A computer system comprising:"

Petitioner asserts that Jiao discloses a "computer system comprising" as graphics processing/processor systems that comprise execution units that can be configured to process a rendering thread, a constant buffer to store constants, and an execution unit control unit to assign threads. Petitioner asserts that a skilled artisan would understand Jiao's graphics processing systems to be a computing system such as graphics processor system 100, which Jiao discloses can be implemented as a computer system. Pet. 36–37 (citing Ex. 1005, 3:31–4:23, 4:54–55, 5:33–57, 6:65–7:18, 8:15–34, 9:39–54, Figs. 1, 2; Ex. 1003 ¶¶ 90–92. These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

b. 15[a]: "a memory;"

Petitioner contends that Jiao discloses a memory as part of its graphics processor system as system memory 124, local memory 106, memory 306, and various caches. Pet. 37 (citing Ex. 1005, 4:49–5:6, 5:33–57, 6:1–48, 6:65–7:27, 7:35–55, 8:15–41, Figs. 1–4; Ex. 1003 ¶ 93). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

c. 15[b]: "a first processing unit;"

Petitioner contends that Jiao discloses a first processing unit as GPU 114 that comprises computational core 204 that includes a pool of multiple execution units to meet the computing requirements of various shader tasks in graphics pipeline 208. Pet. 37–38 (citing Ex. 1005, 5:47–51, 4:49–5:57, 6:1–7:55, Figs. 1–4; Ex. 1003 ¶ 94). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

d. 15[c]: "a bus coupled to the memory and the first processing unit,"

Petitioner asserts that Jiao discloses a bus coupled to the memory and GPU as 512-bit buses coupled to memory 306 and graphics processor 202 (and computational core 204) and to vertex-stream cache and computational core 204 via PCI-E BIU (bus interface unit) 118. Pet. 38 (citing Ex. 1005, 4:49–5:6, 7:19–55, Fig. 1; Ex. 1003 ¶ 95). Petitioner asserts that a skilled artisan would have found it obvious that Jiao discloses a bus coupled to system memory 124 and GPU 114 (and its core 204) via PCI-E BIU 118 and to local memory 106 and the GPU via memory interface unit (MIU) 110 and plural 512-bit buses coupled to memory 306 and graphics processor 202 of GPU 114 (and core 204). *Id.* at 38–39 (citing Ex. 1005, 4:49–5:6, 7:19–55, 8:60–64, Fig. 1; Ex. 1003 ¶ 96). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

e. 15[d]: "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"

Petitioner asserts that Jiao's computational core 204 includes vertex shader 304, geometry shader 306, and pixel shader 310 executed or mapped to core 204 as "engines." Petitioner alternatively asserts that the "engines" are these shaders plus "thread slots" within execution units 420 of EU pool 414, and EU pool control unit 206 stores a plurality of tasks or threads and assigns a thread to an empty thread slot in a respective execution unit 420. Pet. 39–40 (citing Ex. 1005, 5:17–23, 5:47–51, 6:1–7:27, 8:35–41, 9:11–24, 9:39–54, Figs. 1, 3, 4; Ex. 1003 ¶¶ 97–99).

Petitioner asserts that Jiao's vertex shader 304, geometry shader 306, and pixel shader 310 are "engines" because they receive high level graphics tasks from driver software 128 in system memory 124 via EU pool control unit 206 and produce low level shader tasks that are mapped to threads of the execution units 420 of computational core 204 by EU pool control unit 206, which assigns vertex shader, geometry shader, and pixel shader tasks to threads of various execution units 420. Petitioner also contends that Jiao's system memory 124 comprises driver software 128 that communicates instruction sets or commands through CPU 126 to registers in GPU 114 and an applications programming interface (API) space 130, and vertex shader 304 in computational core 204 implements vertex changes in response to commands from driver software 128 to transform objects. Pet. 40–44 (citing Ex. 1005, 1:33–46, 2:8–14, 5:17–23, 5:47–51, 6:65–67, 7:57–62, 8:65–67, 9:10–13, 9:43–49; Ex. 1003 ¶¶ 100–104; Ex. 1008, 36–38).

Petitioner asserts that Jiao's EU pool control unit 206 is a "scheduling module" that sends tasks to the vertex, geometry, and pixel shaders by acting as a global scheduler for tasks and assigning threads in execution units 420 and by assigning vertex shader, geometry shader, and pixel shader tasks to threads of various execution units 420. Petitioner also asserts that EU pool control unit 206 is associated with command stream processor 406, which is a "second processing unit." Pet. 44–45 (citing Ex. 1005, 3:48–60, 4:10–23, 6:65–7:27, 9:15–24, 9:39–54, 10:55–11:46, 14:13–23, Figs. 3, 4; Ex. 1003 ¶ 105). *Alternatively*, Petitioner contends that driver software 128 is a "scheduling module" that communicates instructions and commands via CPU 126, which is a "second processing unit," to registers in GPU 114. Pet. 45–46 (citing Ex. 1005, 5:8–13, 5:17–23, 7:56–8:14; Ex. 1003 ¶ 106).

Petitioner asserts that the shaders load constants to render context and graphics data before executing a task, and shader constants are provided by a constant buffer of constant buffer system 200 or through an EU pool control logic. Pet. 46–47 (citing Ex. 1001, 1:53–57, 3:3–7; Ex. 1005, 1:50–58, 6:1–7:55, 8:65–9:24, 12:33–52, 12:63–13:6, Figs. 3, 4; Ex. 1003 ¶¶ 108, 109).

f. Are Jiao's shader programs "engines"?

Patent Owner argues that Jiao's vertex shader, geometry shader, and pixel shader are shader programs, which the '381 patent describes as parts of tasks that define a sequence of instructions of threads that are parts of tasks. Prelim. Resp. 45. Patent Owner argues that the '381 patent describes a GPU executing a plurality of threads (sequence of instructions) where each thread may comprise a shader program such as a geometry shader, a pixel shader, a vertex shader, or the like, and it equates threads with shader programs. *Id.*

The '381 patent describe the claimed "engines" being embodied on a single shader core and configured to receive and independently execute tasks on shader core 214. Ex. 1001, 4:45–51, 6:42–54, 7:1–19, 10:33–55, Fig. 2. Vertex analyzer 208 identifies shader programs associated with a task and *schedules when to launch each shader program* in shader core 214, which executes the shader programs. *Id.* at 3:20–31, 7:42–48, 8:63–65, *id.* at 1:46–66 (multiple shader programs may be *issued* to the shader core).

Jiao's graphics processor 202 comprises computational core 204 and its graphics pipeline 208 that includes vertex 304, geometry 306, and pixel shader 310 *modules/components mapped to core 204* to generate tasks for execution units 420 in core 204. Ex. 1005, 5:33–51, 6:1–64, 7:56–67, 8:8–12, 9:25–47. The shaders are components of graphics pipeline 208 and elements of core 204. They are not *launched* as in the '381 patent.

Rather than being launched/issued like the shader programs discussed in the '381 patent, Jiao's vertex shader 304, geometry shader 306, and pixel shader 310 are embodied on computational core 204 to process tasks using EUs 420 of EUP 414 like the claimed engines. "The invention is not the language of the [claim] but the subject matter thereby defined." *ADASA Inc. v. Avery Dennison Corp.*, 55 F.4th 900, 913 (Fed. Cir. 2022). These shader components of graphics pipeline 208 and core 204 include internal registers and various caches for configuring their fixed function graphics hardware units for operations in computational core 204. Ex. 1005, 6:2–50, Figs. 3, 4.

g. Do Jiao's shaders "receive a plurality of tasks"?

Patent Owner argues that claim 15 requires "a plurality of engines configured to receive . . . a plurality of tasks" but Jiao's shader programs do not receive tasks. Prelim. Resp. 44. Patent Owner argues that Petitioner's contention that the vertex shader 304, geometry shader 306, and pixel shader 310 of Jiao are "engines" fails because "Jiao states that the alleged engines are 'shader programs'" and the '381 patent teaches that "shader programs" are part of "tasks" rather than "engines" that receive tasks. *Id.* at 44–45.

Jiao's system uses driver software 128 to communicate instructions or commands through CPU 126 to registers in GPU 114. Ex. 1005, 5:17–23. Responsive to commands from driver software 128, such as draw a triangle, information is passed to *vertex shader in computational core 204* to *perform tasks of vertex transformations*. *Id.* at 7:57–67. Jiao's shaders 304, 306, 310 also generate tasks/functions to perform tasks that they receive from driver software 128 in EUP 414. *See id.* at 5:33–57, 6:1–50, 7:56–67, 9:15–54. EU pool control unit 206 assigns these tasks to threads slots in EUs 420, which may be parts of engines under the Petition (Pet. 40). *Id.* at 9:15–54.

The ’381 patent describes the knowledge in the art that GPUs receive tasks, instructions, and commands from applications via drivers that translate commands received from API communications to instructions that GPUs can execute. Ex. 1001, 1:33–46. Jiao’s shaders 304, 306, 310 are capable of performing different tasks, and they perform particular tasks on EUs 420 of EUP 414 based on the commands that the GPU receives from CPU 126 (via driver software 128) and the shaders receive via EU pool control 206. *See* Ex. 1005, 6:1–64, 7:6–65, 9:11–47.

h. Do Jiao’s shaders receive tasks from driver software 128 via EU pool control unit 206?

Patent Owner argues, “the only things the shader programs receive[] are data” in Figure 3 where the only inputs to shader programs 304, 306, 310 are textures, constants, and what is sent from input assembler 302. Prelim. Resp. 47–48. Yet, the shaders use the data to perform tasks and operations (Ex. 1005, 6:1–64) that they receive from driver software 128 via EU Pool control 206 with necessary data provided to computational core 204 (*id.*, *id.* at 7:19–67). Shaders 304, 306, 310 execute these tasks on EUs 420 of EUP 414 by EU Pool Control 206 assigning the tasks to threads in EUs 420. *Id.* at 5:17–57, 7:56–8:34, 9:7–54. “The computational core 204 comprises a pool of multiple execution units to meet the computing requirements imposed by shader tasks associated with various shader programs, including a vertex shader, geometry shader, and/or pixel shader, processing data for the graphics pipeline 208.” *Id.* at 5:47–51 (cited in Pet. 39, 40, 41, 44). Shaders 304, 306, 310 perform received tasks by generating tasks that are assigned to threads of EUs 420 by EU pool control unit 206. *Id.* at 9:11–54, 7:56–8:34. Shaders perform the received tasks in shader core EUP 414.

Patent Owner's declarant, Dr. Hart, testifies that "Jiao itself confirms the only things the shader programs receives are data" in Figure 3. Ex. 2001 ¶ 71. Dr. Hart testifies that Jiao's shader programs do not receive tasks just because they output tasks because they "generate" the tasks that they output and the output tasks are not received from a scheduling module. *Id.* ¶ 74.

Jiao's shaders perform certain tasks (of the many they are capable of performing) based on the tasks that they receive from driver software 128 via EU Pool control 206. Ex. 1005, 6:1–64, 7:19–67. Thus, in response to a command from driver software 128 to draw a triangle, information is passed to vertex shader 304 in computational core 304 to *implement this task as vertex transformations* in graphics pipeline 208. *Id.* at 7:56–65. Shaders 304, 306, 310 perform the received tasks by generating subtasks for EUs 420 to perform. *Id.* at 5:17–57, 7:56–8:34, 9:7–54. Shaders 304, 306, 310 thus "generate tasks for execution units [420] in the computational core 204" and EU pool control unit 206 delivers the tasks to threads in execution units 420 of EUP 414 of computational core 204. *Id.* at 9:49–50, 9:11–47. "EU pool control unit 206 is responsible for assigning vertex, shader, geometry shader, and pixel shader tasks to threads of the various execution units 420." *Id.* at 9:43–46. "As individual tasks are generated, the EU pool control unit 206 handles the assigning of those tasks to available threads within the various execution units 420." *Id.* at 9:39–41. "The EU pool control unit 206 functions as a global scheduler for the tasks and assigns appropriate threads in the execution units 420 using a data-driven approach (e.g., vertex, pixel, geometry packets in the input)." *Id.* at 9:15–18.

At this stage, we find that Petitioner has established that Jiao discloses a plurality of engines that receive a plurality of tasks in element 15[d].

i. 15[e]: "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."

Petitioner asserts that Jiao's computational core 204 comprises EUP 414 and EUs 420a–h that process instructions. Petitioner asserts that EUP 414 is a shader core "configured to receive the plurality of tasks from at least a vertex shader, a geometry shader, and/or a pixel shader (i.e., the "engines") in graphics pipeline processing" and "execute multiple, simultaneous tasks based on respective state data." Petitioner asserts that a skilled artisan "would know that Jiao's high-level tasks (commands) are processed by the vertex shader 304, geometry shader 306, and pixel shader 310 and mapped to low level threads running on the execution units of the computational core by the EU pool control unit." Pet. 47–48 (citing Ex. 1005, 1:43–58, 3:48–60, 6:49–7:18, 8:15–9:14, 9:25–54, 9:65–10:8; Ex. 1003 ¶¶ 110–112).

Patent Owner asserts that "the execution units 420 in EUP 414 receive something that did not even exist until the shader programs generated them [and] cannot be '***the plurality of tasks***,'" engines received. Prelim. Resp. 64.

The '381 patent indicates a driver translates tasks into instructions for a GPU to execute. Ex. 1001, 1:38–48, 9:53–61. Engines retrieve tasks from buffers and provide them to input logic, which processes tasks before issuing them to shader core 214 of GPU 110. *Id.* at 7:1–21, Fig. 2. "[I]nput logic includes graphics pre-processing logic (which prepares graphics-processing tasks for issuance to shader core 214)." *Id.* at 7:25–32. Vertex analyzer 208 provides tasks with data needed by shader programs that are launched in shader core 214 to perform these *processed tasks*. *See id.* at 7:42–62.

Interpreted in light of the '381 patent specification, this language does not require the tasks that are provided to the shader core to be identical to the tasks received by the engines. The tasks may be processed to execute in the shader core. Thus, Petitioner asserts that the shader core in Jiao receives tasks from shaders 304, 306, 310 in a format that it can process and execute.

> In Jiao, the vertex shader 304, geometry shader 306, and pixel shader 310, which constitute the claimed "engines" *receive graphics (high level) tasks* from the driver software 128 residing in the system memory 124 via the EU pool control unit 206 and *produce (low level) shader tasks that are then mapped to threads of the various execution units 420* by the EU pool control unit 206.

Pet. 41 (emphasis added). Petitioner contends that the shaders *process* high-level tasks, and EU pool control unit 206 maps the tasks to low level threads running on the execution units of the core. *Id.* at 48.

Accordingly, on this record, we find that Petitioner has established a reasonable likelihood that claim 15 is unpatentable over Jiao.

3. Claims 16–20

Petitioner provides detailed contentions for claims 16–20 including that Jiao executes various shader tasks as they are delivered to EUs 420 for claim 16, each EU 420 may store tasks and threads in thread slots as queues in claim 17, tasks of different latencies are processed on fast access and slow access paths 506, 508 for claim 18, the tasks are graphics-processing and general-compute tasks in claim 19, and the shader core receives tasks based on priority in claim 20. Pet. 48–52. These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

Accordingly, on this record, we find that Petitioner has established a reasonable likelihood that claims 16–20 are unpatentable over Jiao.

F. Ground 3: Alleged Obviousness Over Stuttard

Petitioner asserts that claims 15–20 are unpatentable under 35 U.S.C. § 103 over Stuttard. Pet. 3, 52–63.

1. Stuttard (Ex. 1006)

Stuttard discloses a graphics data processing system 3 as illustrated in Figure 1, which we reproduce below.

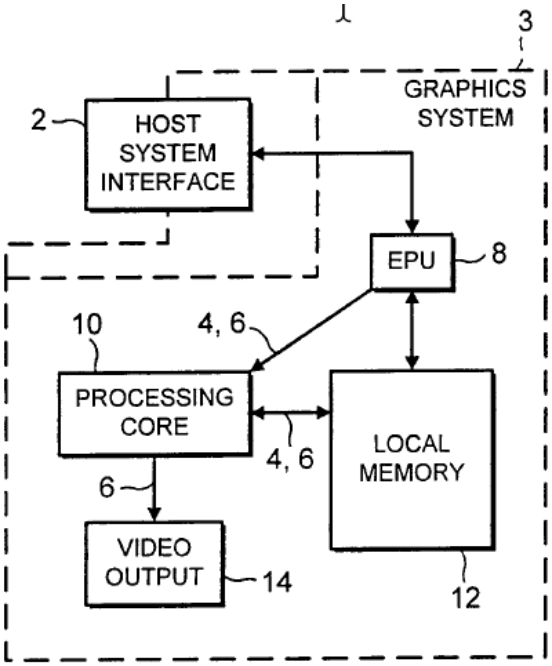

FIG. 1

Figure 1 shows system 3 comprising processing core 10 that provides graphics processing capability for the system. Ex. 1006, 4:21–34, 7:25–26. Embedded processor unit (EPU) 8 controls overall functions of the graphics processor and interfaces with the host system. *Id.* at 4:25–30.

Processing core 10 includes control units such as thread manager 102, array controller 104, channel controller 108, and binning unit 1069 that control operation of processing blocks that perform the graphics processing. Ex. 1006, 8:1–7. We reproduce Figure 3 of Stuttard below to illustrate these control units of processing core 10.

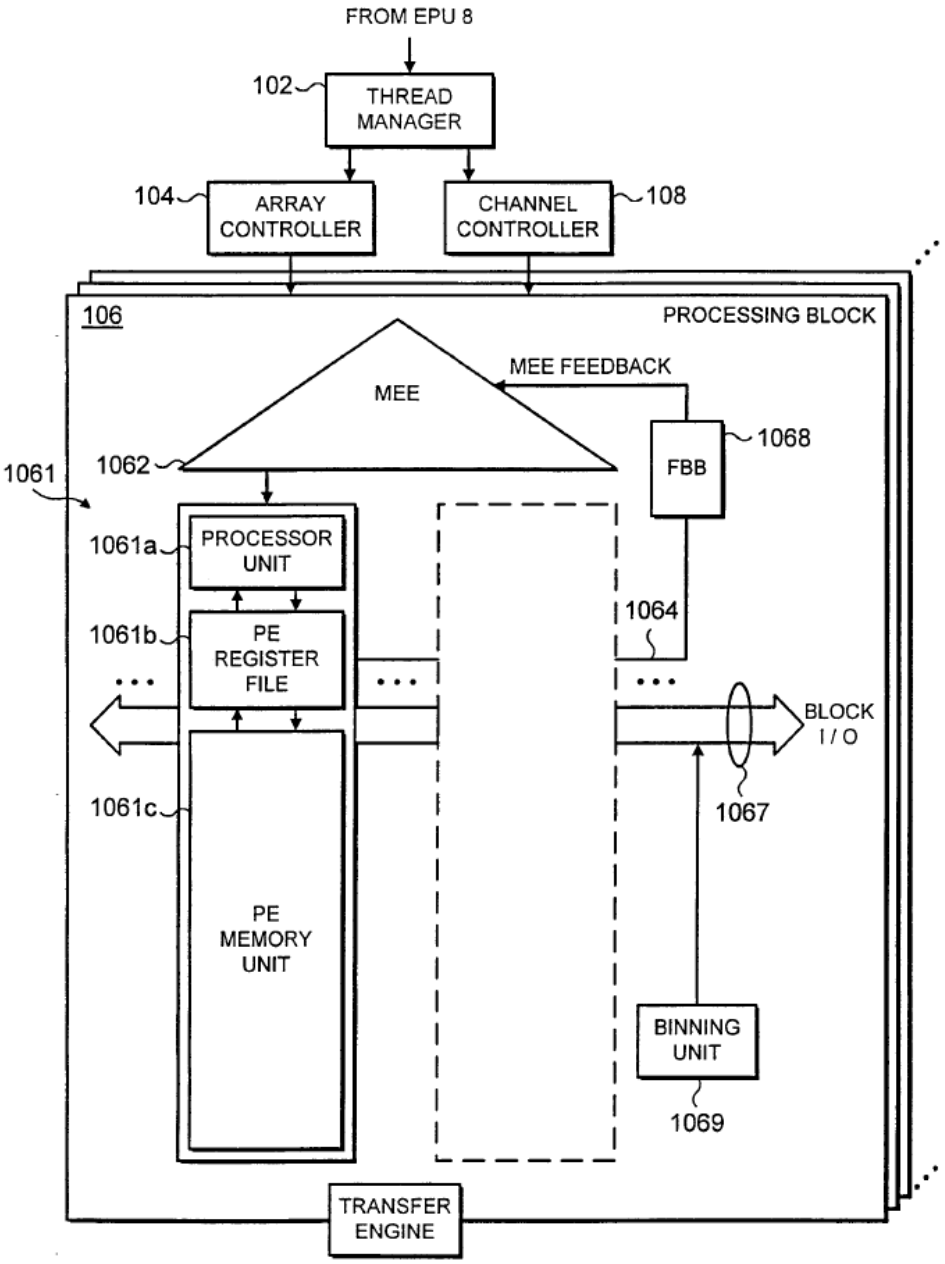

FIG. 3

Figure 3 of Stuttard depicts thread manager 102 connected to EPU 8 to receive control signals and fetch instructions that thread manager 102 provides to array controller 104 and channel controller 108 to control the operation of processing blocks 106 and their array 1061 of the processor elements (PEs). Ex. 1006, 8:17–31.

2. Independent Claim 15

a. Elements 15[preamble], 15[a], 15[b], and 15[c]:

Petitioner contends that Stuttard discloses the preamble, a memory, a first processing unit, and a bus in elements 15[pre]–15[c] as local memory 12 in graphics system 3, first processing core 10 of graphics data processing system 3, and primary bus 4. Pet. 52–55. These contentions, which Patent Owner does not contest at this stage, are supported by record evidence.

b. 15[d]: "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"

Petitioner asserts that Stuttard discloses binning unit 1069, channel controller 108, array controller 104, and thread manager 102 components (cache memory unit 1024, thread processors 1026, and status block 1030) as engines that receive tasks from scheduling modules such as instruction fetch unit 1023 or thread scheduler 1025. Alternatively, Petitioner asserts that the engines receive tasks from EPU 8 as a "scheduling module" associated with a host system "second processing unit." Pet. 55–57 (citing Ex. 1006, 7:25–8:7, 8:17–27, 10:7–17; Ex. 1003 ¶¶ 123–125).

c. Are elements 1023 and 1025 scheduling modules?

Instruction fetch unit 1023 and thread scheduler 1025 are units of thread manager 102 in processing core 10, which is the first processing unit. Thus, they are not "associated with a second processing unit." *See* Prelim. Resp. 71–73; Ex. 2003, 175; Ex. 1006, 8:1–7, 11:3–10, Figs. 3, 4; Ex. 1002, 183 (arguing that a prior art scheduler in the same processing unit as the engines that it controlled is not "associated with a second processing unit").

d. Is EPU 8 a scheduling module?

Thread manager 102 receives control signals from EPU 8 and uses the signals to retrieve instructions, which are provided to array controller 104 and channel controller 108, which transfer the control signals to processing blocks 106 and processing array 1061 to execute. Ex. 1006, 8:17–27, Fig. 3. The '381 patent also discloses an OS scheduler placing tasks in command buffers for virtual engines to retrieve and process the tasks by issuing tasks to shader core 214 for execution. *See* Ex. 1001, 2:8–19, 5:4–13, 6:65–7:62, 10:22–41; Pet. 56–57. Thus, EPU 8 operates as a scheduling module.

e. Do Stuttard's engines receive tasks?

Patent Owner argues that claim 15 requires each engine to receive the tasks, and "Petitioner has not shown that each of its alleged engines receives a task." Prelim. Resp. 76. Patent Owner asserts that primitive data received by Binning Unit 1069 and status information received by Status Block 1030 are not tasks. *Id.* at 76–77. Patent Owner argues that only thread processor 1026 is connected to Thread Scheduler 1025, the first alleged scheduling modules. *Id.* at 77–79. Patent Owner asserts that none of the engines receives anything from EPU 8 because only Fetch Unit 1023, which is not an engine, receives alleged tasks from EPU 8. *Id.* at 79. We disagree.

Thread manager 102 uses control signals from EPU 8 to retrieve instructions that are provided to array controller 104 and channel controller 108, which transfer the control signals to processing blocks 106 and array 1061 of the processing elements to execute. Ex. 1006, 8:17–27, 12:20–22, 26:3–7. Thread processors 1026 issue core instructions to keep processing blocks 106 active. *Id.* at 11:27–12:7. Thus, commands provided by EPU 8 are received by the asserted engines 102, 104, 108, 1026. *See* Pet. 56.

f. 15[e]: "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."

Petitioner contends that Stuttard discloses parallel threads assigned to different tasks at the same time on processing blocks 106 as a shader core. Pet. 59–60 (citing Ex. 1006, 9:11–20, 10:7–17, 11:19–25, 11:27–12:7, 12:30–13:7, 13:23–30, 17:28–18:8, Figs. 3, 8; Ex. 1003 ¶ 129). These contentions, which Patent Owner does not contest at this stage, are supported by record evidence. Threads may perform a single task or multiple tasks in "multitasking" or tasks by priority. Ex. 1006, 10:7–17, 11:16–25.

Accordingly, on this record, we find that Petitioner has established a reasonable likelihood that claim 15 is unpatentable over Stuttard.

3. Claims 16–20

Petitioner contends that Stuttard discloses processing elements that process first and second tasks in claim 16, using queues in claim 17 for low and regular latency tasks in claim 18, graphics and general compute tasks in claim 19, and receiving tasks based on priority in claim 20. Pet. 60–63. These contentions, which Patent Owner does not dispute at this stage, are supported by sufficient record evidence.

Accordingly, on this record, we find that Petitioner has established a reasonable likelihood that claims 16–20 are unpatentable over Stuttard.

III. CONCLUSION

For the reasons discussed above, we are persuaded that Petitioner has demonstrated a reasonable likelihood of prevailing in establishing that at least one of claims 15–20 of the '381 patent is unpatentable.

IV. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that *inter partes* review of the ’381 patent is hereby instituted as to claims 15–20 on all grounds set forth in the Petition;

FURTHER ORDERED that pursuant to 35 U.S.C. § 314(c) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial on the grounds of unpatentability authorized above; the trial commences on the entry date of this decision; and

FURTHER ORDERED that the trial will be conducted in accordance with a corresponding separately issued Scheduling Order.

PETITIONER:

Jeffrey Johnson
Christopher Higgins
ORRICK, HERRINGTON & SUTCLIFFE LLP
3J6PTABDocket@orrick.com
OCHPTABDocket@orrick.com

Steven Baik
WHITE HAT LEGAL
sbail@sidley.com

PATENT OWNER:

William Meunier
Michael Renaud
Adam Rizk
MINTZ, LEVIN, COHN, FERRIS, GLOVSKY and POPEO, P.C.
WAMeunier@mintz.com
MTRenaud@mintz.com
ARizk@mintz.com