

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,
v.

ADVANCED MICRO DEVICES, INC.
Patent Owner.

Case No. IPR2023-00687

U.S. Patent No. 8,854,381

**PETITIONER'S REPLY TO PATENT OWNER'S
RESPONSE TO PETITION**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	CLAIM CONSTRUCTION	1
A.	“task”	1
B.	“engine” and “each of the plurality of tasks”	3
C.	“configured to”	3
III.	THE CLAIMS ARE RENDERED OBVIOUS BY WILT	3
A.	All Claims.....	3
1.	Wilt Discloses a Shader Core that Simultaneously Executes Two Tasks	3
2.	A Shader Core Comprised of Multiple PPUs Discloses and Renders the Claims Obvious.....	7
3.	Wilt’s Engines are Not Within the Shader Core.....	8
4.	The Petition Clearly Identified Which Components in Each PPU Are The “Engines” And The “Shader Core”	8
B.	Claim 19	9
1.	Wilt Expressly Discloses Its Multiple PPUs Simultaneously Executing Graphics and General-Compute Tasks	9
2.	Each “Single PPU” Can Simultaneously Execute Graphics and General-Compute Tasks.....	10
C.	Claim 20	11
IV.	THE CLAIMS ARE RENDERED OBVIOUS BY JIAO.....	11
A.	All Claims.....	12
1.	Jiao Discloses and Renders Obvious the Claimed “Plurality of Engines”	12
2.	Jiao’s Scheduling Module is Not “Associated With” the First Processing Unit.....	14
3.	Jiao’s Shader Core Receives the Plurality of Tasks	15
4.	Jiao Discloses Receiving and Simultaneously Executing At Least Two Tasks	16
B.	Claim 19	17

C.	Claim 20	19
V.	THE CLAIMS ARE RENDERED OBVIOUS BY STUTTARD	20
A.	All Claims.....	20
1.	Stuttard Discloses Receiving and Simultaneously Executing The Claimed “Tasks”	20
2.	Stuttard’s Engines are Not Within the Shader Core	23
3.	Stuttard’s EPU 8 is a Scheduling Module as Claimed	23
4.	The Claims Do Not Require “Parallel” Engines or that Each Engine Perform Every Claim Function	24
B.	Claim 19	24
C.	Claim 20	25

PETITIONER'S EXHIBIT LIST

Exhibit #	Reference Name
1001	U.S. Patent 8,854,381 to Lefebvre et al.
1002	Prosecution History of U.S. Patent 8,854,381
1003	Declaration of Massoud Pedram, Ph.D.
1004	U.S. Patent No. 8,151,095 to Wilt ("Wilt")
1005	U.S. Patent No. 8,319,774 to Jiao et al. ("Jiao")
1006	PCT Pub. No. WO00/62182A2 to Stuttard et al. ("Stuttard")
1007	Curriculum Vitae of Massoud Pedram, Ph.D.
1008	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318 (Order No. 31, Oct. 18, 2022)
1009	Deposition Transcript of Dr. John Hart
1010	Declaration of Dr. John Hart dated August 15, 2022
1011	ITC claim construction ruling (Order No. 31)
1012	Declaration of Dr. Massoud Pedram
1013	AMD Brief (177 Proceeding related to 1318 ITC investigation)

TABLE OF AUTHORITIES

Page(s)

CASES

KSR Int’l Co., v. Teleflex Inc., 550 U.S. 398 (2007).....	13
SanDisk Corp. v. Kingston Tech. Co., 695 F.3d 1348 (Fed. Cir. 2012)	8
Textron Innovations, Inc. v. American Eurocopter Corp., 498 Fed. Appx. 23 (Fed. Cir. 2012).....	7

OTHER AUTHORITIES

37 C.F.R. § 42.6(e).....	28
--------------------------	----

I. INTRODUCTION

The Petition shows that claims 15-20 of U.S. Patent No. 8,854,381 are unpatentable.

II. CLAIM CONSTRUCTION

A. “task”

Dr. Hart previously testified that “‘task’ means ‘a unit of processing work, comprising one or more threads and/or shader programs.’” Ex. 1010, ¶76. Dr. Hart now contradicts his prior testimony, and testifies, “‘task’ should be construed as ‘a unit of processing work decomposable into two or more related threads.’” Ex. 2006, ¶29. Dr. Hart has offered no opinions under any other construction or using the plain and ordinary meaning of “task,” which the ITC determined to be “a unit of processing work or portion of an application to be executed.” Ex. 1011, 36-38; Ex. 1009, 43:5-44:6, 47:7-16.

Dr. Hart admitted that a processor can only execute threads. Ex. 1009, 67:13-68:9. But Dr. Hart testified that “if you have a shader core that’s only receiving threads, it’s not receiving a task.” *Id.* 73:20-21. Then Dr. Hart testified that even if a shader core receives all the decomposed threads of a task, it *still* has not received the “task.” *Id.* 60:15-61:10, 65:15-66:17, 70:14-71:2, 74:16-75:7, 82:4-16. That is because Dr. Hart has also imported a knowledge requirement into claim 15: even if a shader core receives all threads decomposed from a task, the

shader core *still* has not received a “task” unless it also “knows” that the threads were decomposed from a single task. *Id.* 60:2-61:10, 65:15-66:17, 72:19-75:7, 81:3-16, 180:7-13. For example:

Q. If that shader core receives all of the decomposed threads from a task, it may still not know it’s receiving a task, and, therefore, it would not be receiving a task under your interpretation, correct?

A. Right.

Id. 72:19-73:5. Consequently, even though Wilt, Jiao, and Stuttard expressly disclose receiving and processing “tasks” and executing corresponding threads, Dr. Hart’s opinion is that *none* of the references disclose the claimed “tasks”—he only found “tasks” in the ’381 patent. *Id.* 178:17-179:7. *See also*, Ex. 2006, ¶¶ 70-71, 116, 135. Because “task” is central to every claim limitation in dispute, Dr. Hart’s improper construction of “task” nullifies most of ATI’s arguments and eviscerates ATI’s Response.

In the ’381 patent, a “task” may be one or more threads or shader programs. Ex. 1011, 36-38; Ex. 1001, 1:47-2:37; 6:45-47, 7:42-43 (“each processing task ... *may* include a plurality of individual processing threads”). Only threads or instructions are executed in a processing core, and each thread has a “context” that includes state data. *See* Ex. 1001, 1:47-2:37. While AMD repeatedly states, “a thread is not a task,” a task can consist of a single thread and a thread executing in a processor may correspond to a higher-level task. Ex. 1012 ¶76.

Even if adopted, the prior art asserted all show tasks decomposable into two or more related threads because under Dr. Hart’s reasoning, all tasks would decompose into two or more related threads. *Id.* ¶3. It is obvious that graphics processing would include tasks complex enough to decompose into two or more related threads. *Id.*

B. “engine” and “each of the plurality of tasks”

ATI’s arguments were all rejected at the ITC. Ex. 2003, 97-98, 153, 174. As the ITC held, the plain reading of claim 15 is that “the plurality—*as a group*—do the receiving,” and “it is the entire group of engines which loads the state data—to be evaluated *as a group*.” Ex. 2003, 97, 174; *cf.* Ex. 1001, 4:48-51 (describing only one embodiment). Consistent with this plain reading of the claim, claim 15 only requires the shader core receive the plurality of tasks from *one* of the plurality of engines, not each.

C. “configured to”

Petitioner agrees that “configured to” implies a system that is designed to perform the recited function. Petitioner’s evidence shows the prior art systems are configured as claimed, or alternatively, the claimed configuration is obvious.

III. THE CLAIMS ARE RENDERED OBVIOUS BY WILT

A. All Claims

1. Wilt Discloses a Shader Core that Simultaneously Executes Two Tasks

ATI relies on its new construction of “task” to argue that Wilt’s PPU’s do not receive “tasks” because they receive “commands” that allegedly correspond with single “threads.” This construction is wrong, *supra* at Section II.A. Wilt expressly discloses that “[c]ores 208 receive processing tasks to be executed via a work distribution unit 210, which receives commands defining processing tasks from a front end unit 212.” Ex. 1003, ¶¶47, 67-68; Ex. 1004, 4:1-23.

ATI argues the shader core does not execute the “tasks” received by the engines, if those tasks are subsequently broken down into smaller units for execution in the shader core. This directly contradicts ATI’s position in the ITC:

ATI has firmly taken the position ... that working on low-level activities qualifies as working on the high-level tasks those low-level activities derive from ... “the ’381 Patent, and claim 15 in particular, allows the shader core to receive tasks ... in a different form of expression, from the tasks received by the engines.”

Ex. 2003, 163 (quoting ATI’s brief). The ’381 patent discloses that “[t]o perform a single graphics-processing task, a GPU may execute a plurality of different threads.” Ex. 1001, 1:47-2:7; 6:42-54. A POSITA would have understood that “task” is a high-level program construct, which may result in one or more threads being executed. Ex. 2007, 17:2-20:18. Once it is recognized the “task” received can be a higher-level construct of the threads executed in the shader core, and that

some “tasks” may correspond to a single thread, ATI’s arguments that Wilt does not disclose simultaneously executing different tasks fail in their entirety.

Wilt discloses simultaneous execution of at least two tasks in the shader core, under two alternative theories. Pet., 10-12, 24-25, 28-31. Under the “single PPU theory,” the “engines” are the front-end unit 212, work distribution unit 210, core interface 303 within a single PPU 202 and the “shader core” is the set of cores 208. *Id.* The cores are organized in the SIMT fashion, which means that concurrent threads running on a single PPU can “follow divergent execution paths,” *i.e.*, threads in each core 208 can be executing two or more different instruction streams corresponding to different tasks at the same time. Ex. 1004, 6:2-10.

Under the “multiple PPU theory,” the collection of engines across multiple PPUs constitute the claimed “engines,” and the set of all cores 208 within all PPUs 202 of the parallel processing system 112 correspond to the claimed “shader core.” The cores receive processing tasks via a work distribution unit and can be programmed (independently) to execute two different types of tasks simultaneously, *e.g.*, “PP subsystem 112 may include one or more PPUs 202 that operate as graphics processors and one or more other PPUs 202 that are used for general-purpose computations.” Pet., 10-12, 24-25, 28; Ex. 1004, 4:1-23, 33-45, 46-60, 8:56-67; Ex. 1003, ¶48.

AMD argues that the threads being simultaneously executed do not necessarily come from different tasks. But Wilt discloses that in its “parallel processing subsystem” its cores “can be programmed to execute processing tasks relating to a wide variety of applications.” *Id.* at 4:33-40. It is inherent, or at least obvious, that different tasks may be executed simultaneously, whether in a single PPU or multiple PPUs.¹

Finally, AMD attempts to argue that Wilt’s system is not “configured to” receive a plurality of tasks and execute tasks concurrently. Wilt expressly discloses that each PPU has multiple programmable parallel cores and an unlimited number of “parallel processing units (PPUs)” in its “parallel processing subsystem.” Pet., 10-12, 24-25, 28; Ex. 1004, 4:1-23, 33-45. A POSITA would understand that Wilt’s system is designed to perform different tasks simultaneously to increased core utilization, lower end-to-end latency, and higher processing throughput. Ex. 1012 ¶20.

¹ Dr. Pedram testified that a task can be comprised of a single thread. Ex. 2007, 11:7-16 (Q. Okay. So a task can be composed of multiple threads; right? A. ***Or a single thread*** ... there’s no requirement that a task results in multiple threads. ***Task could be a single thread.***”) Whether any specific thread can be traced to a specific parent task has no relevance.

2. A Shader Core Comprised of Multiple PPU's Discloses and Renders the Claims Obvious

AMD argues that the “multi-PPU” alternative in Wilt does not satisfy the claim terms “a plurality of engines” or “a shader core.” This is incorrect.

First, AMD argues that the front end, work distribution unit, and core interface of a single PPU cannot collectively constitute an “engine.” POR 45. Multiple components can be a single claimed unit. *See Textron Innovations, Inc. v. American Eurocopter Corp.*, 498 Fed. Appx. 23, 30 (Fed. Cir. 2012) (structural limitations are generally not construed to require a single unitary structure). These three units in each PPU, collectively, receive a plurality of tasks from a scheduling module associated with a CPU and load state data associated with each of the tasks. Pet., 26-27; Ex. 1003, ¶73. AMD does not argue otherwise, POR, 45-46, and Dr. Hart agreed that an “engine” can be anything that performs the claimed functions. Ex. 1009, 116:7-117:11. Moreover, the ITC accepted AMD’s argument that in the Accused Products, a “Job Slot” outside the shader core and a “Frontend” inside the shader core could constitute a single “engine.” Ex. 2003, 146, 151, 166.

Second, AMD argues that the processing cores across multiple PPUs cannot be the claimed “shader core” because, *inter alia*, the cores “cross PPU boundaries,” do not “communicate with each other,” and only receive commands from their own engines. But these are not claim requirements. The ’381 patent defines “shader

core” as an “*an array* of processing elements,” Ex. 1001 at 1:59-61. The processing cores across the parallel PPU’s satisfy 15[e] by receiving a plurality of tasks from at least one “engine” and executing two tasks simultaneously based on state data. And even if required, Wilt discloses that PPU’s can communicate with each other through the bidirectional communication path 113 and the memory bridge 105. *See* Ex. 1004, 3:49-4:23 and Fig. 3. A POSITA would understand that the “engine” in each PPU can send processing tasks to the processing cores of other PPU’s. Ex. 2007, 42:21-45:6.²

3. Wilt’s Engines are Not Within the Shader Core

AMD argues that Wilt’s “engines” are within the “shader core.” They are not. Under both theories, Petitioner identified the “engines” as components separate from the processing cores 208 inside each PPU (the “shader core”), which cores “execute concurrent tasks” *received from* the engines. Pet., 26-28; ID, 19.

4. The Petition Clearly Identified Which Components in Each PPU Are The “Engines” And The “Shader Core”

² In addition, “a” shader core is presumed to mean “one or more” shader cores. *SanDisk Corp. v. Kingston Tech. Co.*, 695 F.3d 1348, 1360-61 (Fed. Cir. 2012). Even if the processing cores in each PPU were considered a single “shader core,” the “shader cores” in all parallel PPU’s would still satisfy the claims.

Petitioner did not simply rely on each PPU as the claimed “shader core,” as AMD contends. Petitioner explained that a POSITA “would have understood the work distribution unit 210, the front-end unit 212, and the core interface unit 303 ... to be the ‘*plurality of engines*,’” and alternatively, that the same components in each PPU are a single “engine” and the engines in the plurality of PPUs constitute the claimed ‘plurality of engines.’” POR, 26, 27. Petitioner also identified the processing cores 208 as “the claimed ‘processing elements’ of the ’381 patent” that satisfy the *shader core* functionality, *i.e.*, receiving and executing tasks. POR, 7, 11-12, 28-29.

B. Claim 19

1. Wilt Expressly Discloses Its Multiple PPUs Simultaneously Executing Graphics and General-Compute Tasks

AMD admits that PPUs can be used for general-compute, AMD only argues that some PPUs cannot perform graphics tasks *while* other PPUs perform general-compute tasks. AMD is wrong.

Wilt discloses that “PP subsystem 112 may include one or more PPUs 202 that operate as graphics processors and one or more other PPUs 202 that are used for general-purpose computations,” and that the parallel processing subsystem 112 can incorporate both “circuitry optimized for graphics” and “circuitry optimized for general purpose processing.” Ex. 1004, 3:24-31, 4:55-60. Accordingly, Wilt’s system is specifically “designed to” execute a graphics processing task while

executing a general-compute task. AMD's suggestion that any PPUs processing graphics would be idle while other PPUs perform general-purpose computations has no support in Wilt, which teaches that multiple PPUs should be "operated in parallel to process data at higher throughput." *Id.*, 5:41-51. It was at least obvious to configure Wilt's system to simultaneously execute the two types of tasks, to achieve the increased throughput that Wilt's system was designed to achieve. *See, e.g.*, Pet. 10, 28-32, 35.

2. Each "Single PPU" Can Simultaneously Execute Graphics and General-Compute Tasks

AMD asserts that Wilt only discloses using PPUs exclusively for graphics or exclusively for general-compute, and a PPU cannot perform both at the same time. Untrue. Each core 208 can carry out SIMT execution, which "allows different threads" to "follow divergent execution paths." Ex. 1004, 5:61-6:12; Pet., 11-12, 28. Wilt further discloses that each "multithreaded processing core 208" can execute "a number (n0) of threads that concurrently execute." Ex. 1004, 8:56-67. A POSITA would have understood that a single PPU with an unlimited number of "SIMT parallel processing core[s]" could execute at least two different threads corresponding to different tasks at the same time. Ex. 1012 ¶41. Further, because each PPU can do both graphics-processing and general-compute tasks, a POSITA would have understood each PPU was configured to execute those two types in the

“divergent execution paths.” *Id.* At a minimum, such a configuration was obvious.³ *Id.*

C. Claim 20

AMD contends that Wilt’s context swapping does not “necessarily involve or require the use of task priorities.” POR, 37. However, the Petition and Dr. Pedram’s testimony shows exactly that. Ex. 1004, Abstract and 12:4-13:15; Ex. 2007, 60:6-9, 61:20-62:1. For example, Wilt discloses that each task may require “a different PPU context.” Ex. 1004, 11:25-29. Swapping PPU contexts means swapping tasks that make use of their associated contexts, and a POSITA would understand that such context swapping is needed to bring higher priority tasks to the forefront so that they can promptly run on the available compute resources. Ex. 2007, 59:6-62:16; Ex. 1012 ¶¶48-50. AMD’s arguments based on its construction of “task” are without merit, especially given AMD’s broad assertion enforcing Claim 20 that merely prioritizing tasks that have an “active” context satisfies the priority limitation of claim 20. Ex. 1013, 70.

IV. THE CLAIMS ARE RENDERED OBVIOUS BY JIAO

³ This theory was not waived. The Petition discussed how Wilt’s single PPUs can process tasks concurrently and can perform both graphics and general-compute. Pet. 28-32, 35.

A. All Claims

The ITC found claims 15-17 anticipated by Jiao. Ex. 2003 at 146-165.

1. Jiao Discloses and Renders Obvious the Claimed “Plurality of Engines”

The Petition shows that a POSITA would have understood Jiao’s shaders (304, 306, 310) to be “engines” as recited in claim 15. Pet., 39-47. Alternatively, Petitioner asserts that the shaders plus the thread slots within execution units are the claimed “engines.” *Id.*

a. Jiao’s Shaders (“Engines”) Receive Tasks

AMD asserts that the “engines” receive only “graphics data,” not tasks. POR, 52-54. AMD admits that an API and driver software “communicates instruction sets or commands through the use of the CPU 126 to registers in the GPU 114.” *Id.*; Ex. 1005, 5:17-23. Jiao further discloses that “individual tasks” are generated by the shaders (engines) and those tasks are assigned to available threads in the execution units. Ex. 1005, 9:35-54.

A POSITA would have understood that the “commands” generated by the driver software in Jiao are the claimed tasks, and the commands must make their way to Jiao’s engines for them to generate the lower-level tasks that are assigned to threads in Jiao’s shader core. Pet., 40-48; Ex. 1005, 5:47-51, 9:41-49. The ITC agreed that viewing Jiao as a whole, “there is no reason to find that the assignments issued by the driver 128, eventually received by VS 304, GS 306, and

PS 310, and then received by EUs 420 are not ‘linked’ or related to another.” Ex. 2003, 149-153 (concluding there was “no merit” to AMD’s position that the shaders do not receive tasks and “it cannot be seriously argued” that the engines do not receive commands that prompt the generation of lower-level tasks).

As the Board recognized, Jiao’s engines perform operations “[r]esponsive to a command from the driver software” and “generate tasks/functions to perform tasks they receive from driver software.” ID, 27-29; Ex. 1005, 5:33-57, 6:1-64, 7:56-67, 9:11-59. Consistent with Jiao, the ’381 patent describes as general knowledge in the art that driver software translates code from an API, provides instructions or “tasks” to the GPU for execution, and in response, the “engines” (shaders) provide tasks that are mapped to the threads of the execution units, none of which would not happen if the engines did not receive the tasks provided by the driver. *Id.*; Ex. 1001, 1:33-57, 9:53-61. *KSR Int’l Co., v. Teleflex Inc.*, 550 U.S. 398, 406 (2007) (obviousness considers prior art as a whole and knowledge of a POSITA).

b. Jiao’s Thread Slots Are Part of the Engines

By designating the thread slots as part of the engines under its alternative theory, Petitioner made clear the thread slots are not part of the shader core. Pet., 40; Ex. 2003, 151. AMD’s remaining argument that storing one thread does not constitute storing a task is based on its erroneous construction of “task.”

c. Jiao's Shaders Are "Engines" And Are Not Part of the Shader Core

Jiao's shaders are the claimed "engines" and are separate from the shader core. As recognized by the Board, Jiao's shaders are "components" of graphics pipeline 208 and elements of core 204. ID, 26-27; Ex. 1005, 5:33-51, 6:1-64, 7:56-67, 8:8-12, 9:25-47, Fig. 3-4. Since tasks from the shaders are delivered to the execution units, they are not "part of" the execution units (the "shader core"). Ex. 1005, 9:25-54. Moreover, claim 15 does not require the claimed engines to be configured in parallel or be independent of one another, as AMD argues. In any event, the EU pool control unit can send tasks from any of the shaders to the execution units, in parallel and independent of each other, *id.*, 9:25-54, Fig. 4, and the '381 patent discloses the engines can be implemented as hardware, software, or a combination. Ex. 1011, 39-41.

2. Jiao's Scheduling Module is Not "Associated With" the First Processing Unit.

Petitioner identified the computational core 204 as the first processing unit, not the GPU. Pet., 37-38; Ex. 1005, Fig. 4. Petitioner then argued that the "scheduling module associated with a second processing unit" may read on the EU Pool Control Unit 206 (associated with command stream processor 406), or alternatively, driver software 128 (associated with CPU 126). Pet., 44-46. Neither the EU Pool Control Unit nor the driver software is *within* the first processing unit

(computational core 204), which is the only subject matter disclaimed in the file history. ID, 9. Nothing in the claims or file history restricts the scheduling module from having a functional relationship with components other than the second processing unit.

AMD’s argument that the EU Pool Control Unit cannot be the “scheduling module” because it is not *within* the second processing unit (command stream processor 406) contradicts the teachings of the ’381 patent. The patent identifies CPU 102 and GPU 110 as the two processing units. Ex. 1001, 10:5-6, Fig. 4. The OS scheduler 310 is the “scheduling module,” but is not depicted as “within” the CPU (“second processing unit”). *Id.*, 6:42-50, 10:22-32; Fig. 4. A POSITA would recognize that the OS scheduler resides in the system memory and is executed on the CPU. Ex. 1012 ¶27.

The ITC also properly concluded that EU Pool Control 206 is “associated with” the second processing unit (command stream processor 406), even though it is outside *both* the “first processing unit”—computational core 204—*and* the “second processing unit.” Ex. 2003, 148, 157-161.

Petitioner’s alternative embodiment is that the driver software constitutes the claimed scheduling module and is associated with the CPU (second processing unit). AMD has not rebutted this alternative.

3. Jiao’s Shader Core Receives the Plurality of Tasks

As the Board agreed, Jiao's shaders (304, 306, 310) ("engines") receive graphics (high level) tasks from the driver software 128 and EU pool control unit 206 and produce (low level) shader tasks that are then mapped to threads of the execution units 420 by the EU pool control unit. Pet. 41; ID, 31.

AMD's arguments are premised on its incorrect construction of "task." In addition, AMD incorrectly argues that tasks cannot be "modified" prior to being received by the shader core based on "input logic" between the engines and the shader core. POR, 59-60. But the '381 patent discloses exactly this. For example, "graphics pre-processing logic," including a vertex analyzer and scan converter, "prepares graphics-processing tasks for issuance to shader core 214." Ex. 1001, 7:30-62, Fig. 2; Ex. 1012 ¶¶29-30. These components process or modify the tasks received by the "engines" before they are sent to the "shader core." Ex. 1001, 7:30-62, Fig. 2; Ex. 1012 ¶¶29-30. Similarly, a POSITA would have understood that in Jiao, high-level tasks are received by the shaders ("engines") and corresponding lower-level tasks are output to the EUs ("shader core"). Ex. 1012 ¶¶29-30.

4. Jiao Discloses Receiving and Simultaneously Executing At Least Two Tasks

AMD's argument that Jiao's shader core does not simultaneously execute tasks again rests on its erroneous construction that executing a "thread" cannot be

executing a “task.” POR, 62. Executing threads or shader programs is executing tasks. Ex. 1012 ¶¶29-30.

AMD is also incorrect that multiple “tasks” in Jiao’s system are only associated with a single context or thread. First, this is not relevant under a proper construction of “task.” Second, Jiao expressly discloses that concurrently running tasks (“shader programs”) may be associated with multiple rendering contexts in each execution unit, and each EU in Jiao is “simultaneously running multiple contexts.” Ex. 1005, 3:53-60, 8:65-9:9:14, 10:14-54; Ex. 1012 ¶31.

B. Claim 19

The ITC found it “more than likely” that Jiao’s GPU could be used for general-compute tasks, which the ITC found insufficient to meet the burden of proof for invalidity. Ex. 2003, 168. It is sufficient to meet the preponderance burden here. Ex. 2003, 168.

AMD contends that Jiao does not expressly disclose general-compute tasks, but fails to rebut Petitioner’s evidence of obviousness. Jiao’s EUP is the “shader core” and simultaneously executes multiple tasks. Pet., 47-48. Jiao discloses “arithmetic logic units and other logic,” which was understood to execute general-compute tasks. Pet., 51. The described functionality of “instruction decoding, operand fetch, branch computation ... among other processing functionality or logic” was understood to be general-compute tasks. *Id.* Moreover, the

computational core 204 was “a software programmable core processing unit” that used programmable shaders (304, 306, and 310) and parallel EUs “configurable for programmed operation.” Ex. 1005, 5:37-41, 9:25-38. The ’381 patent discloses that 3D graphics engines may generate general-compute tasks. Ex. 1001, 7:34-46. AMD agrees, taking the position that “general-compute tasks” include “complex mathematical algorithms” and “can also be used to support computations that are part of graphics processing.” Ex. 1013, 15-16. And Jiao discloses vertex, geometry, and pixel shaders are only *examples* of possible shaders. *Id.* 5:47-57, 6:59-64.

Based on these disclosures, a POSITA would have found it obvious to use Jiao’s programmable core to perform graphics-processing and general-compute tasks simultaneously. Pet. 51; Ex. 1012 ¶43. Dr. Hart admitted that he did not formulate any definition of what constitutes a “general compute task” or an opinion of whether Jiao can execute a general compute task. Ex. 1009, 129:13-134:21, 155:11-16. Dr. Hart also testified that “general-compute tasks” can be used for graphics processing (*Id.* 20:4-13), that video game applications use both general-compute and graphics tasks (*Id.* 129:9-12) and that both graphics and general compute tasks utilize arithmetic operations (*Id.* 40:11-17). Dr. Pedram’s opinion is substantively un rebutted.

Finally, AMD argues the EUDP is not within the shader core. Jiao discloses that EUDP 514 is part of the computational core EUs 420, *i.e.*, the shader core. Ex. 1005, 10:33-39; Ex. 2007, 119:6-19 (EUDP “part of the shader core”).⁴

C. Claim 20

AMD wrongly contends that constants with a priority level do not impact the receipt of tasks by the shader core. Each thread’s rendering context includes “constants with a priority level.” Ex. 1005, Abstract, 2:24-36, claim 1. One example of a priority is the fast path or slow path for loading constants. *Id.*, 10:25-31. “The rendering context ... is passed from the EU pool control unit 206 ***before tasks belonging to this context are commenced.***” *Id.*, 9:5-7. The constants for a rendering context must be received by the EUs first, and “[t]hen, the execution unit 420 is ready to receive tasks that use the new constant context.” *Id.*, 13:32-53, 13:63-14:46. A task is not sent to the execution unit ***until and unless*** its corresponding rendering context has been loaded—including the associated constants—and those constants are loaded based on a priority. *Id.* Thus, Jiao’s shader core is configured to receive the plurality of tasks based on a priority associated with the tasks.

⁴ Dr. Pedram’s cited testimony (POR, 31) related to the shaders/engines of one disclosed embodiment, not the EUP or what would have been obvious.

V. THE CLAIMS ARE RENDERED OBVIOUS BY STUTTARD

A. All Claims

The only limitation of claims 15-20 that the ITC court found missing in Stuttard was the “scheduling module associated with a second processing unit” because it found fetch unit 1023 and thread scheduler 1025 (one alleged “scheduling module”) to be *inside* the “first processing unit.” Ex. 2003, 175. However, the ITC did not consider Petitioner’s alternative argument that Stuttard’s EPU 8 satisfies the claimed scheduling module limitation, which is clearly *outside* the “first processing unit.” ID, 34-35.

1. Stuttard Discloses Receiving and Simultaneously Executing The Claimed “Tasks”

AMD’s arguments are based on its improper construction of “tasks” and its erroneous arguments that threads, a single thread, and/or instructions cannot be a “task.” POR, 64-66.

Stuttard’s plurality of engines (channel controller 108, array controller 104, and thread manager components 1024, 1026, and 1030) and the shader core (processing blocks 106) both receive the claimed “tasks.” Pet., 55-60. EPU 8 (“scheduling module”) sends control signals to the thread manager 102, which uses the signals to retrieve instructions that are issued to array controller 104 and channel controller 108 by thread processors 1026 (each of which are “engines”). Ex. 1006, 11:12-12:7, 26:3-7. Thus, instructions provided by EPU 8 are received

by “engines” 104, 108, 1024, 1026 and make their way to the shader core. The “instruction streams for the processing blocks 106 are known as ‘threads’” and “each thread works cooperatively with other threads to perform a task or tasks,” which supports “multi-threading” and “multitasking.” *Id.*, 10:7-17. Thus, the “engines” and the “shader core” both receive instructions that comprise one or more threads, which can correspond to one or more tasks. As the Board and ITC recognized, these instruction streams meet the plain meaning of “tasks.” *ID*, 35; Ex. 2003, exhibit page 173.

AMD also erroneously argues that Stuttard does not disclose or render obvious simultaneously executing different threads across its eight or more processing blocks. The Board in the co-pending IPR challenging U.S. Patent No. 7,742,053 summarized Stuttard well:

“ATI does not address a fundamental principle of Stuttard, i.e., ***the division of PEs into processing blocks***. ... Although Stuttard provides a ‘particular example’ in which the same instruction stream is supplied to all the processing blocks ... ***Stuttard explicitly discloses that different blocks may be assigned different tasks.***”

IPR2023-00564, Paper 7 at 31-32 (citing Stuttard, 11:19-25) (emphasis added).

Petitioner made the same arguments here. Pet., 16-17, 59-60.

Stuttard discloses that “each thread [threads 0 to 7] may be assigned to any one of a number of tasks.” Ex. 1006, 11:16-35. AMD argues that this only applies

to threads in the cache of the thread manager. But “the thread manager includes one thread processor 1026 for each thread,” and the “thread processors 1026 control the issuance of core instructions from the thread manager so as to *maintain processing of simultaneously active program threads, so that each of the processing blocks 106 can be active for as much time as possible.*” *Id.*

Stuttard further discloses that threads may perform a (single) task or (multiple) tasks. *Id.*, 10:7-12, 23:35-25:17. Stuttard’s explicit teaching that it can run 8 parallel threads that are assigned to different tasks at the same time directly refutes AMD’s argument. Ex. 1006, 11:19-25. The example of supplying the same instruction stream to all of the processing blocks is described by Stuttard as only one “particular example.” *Id.*, 11:33-35.⁵ AMD’s other cites to Stuttard are also directed to particular embodiments and fail to consider Stuttard as a whole.

The undisputed evidence—under a proper construction of “task”—is that a POSITA would have understood Stuttard discloses a system configured to execute two or more tasks simultaneously. Ex. 1012 ¶¶36-39. Alternatively, a POSITA would understand such a configuration to be obvious due to (1) the explicit

⁵ ATI’s suggestion that the purpose for Stuttard’s partitioning is to operate on different regions of the display or for redundancy has little practical benefit. Ex. 1012 ¶38.

disclosures of Stuttard (Ex. 1006, 11:19-25), (2) the benefits of such a configuration, including higher utilization of the processing elements (*Id.*, 1:24-2:10), and (3) the knowledge in the art of how parallel processing including multi-threading and multi-tasking can improve the overall system performance and computational throughput (Ex. 1006, 1:11-18). Ex. 1012 ¶¶36-39.

2. Stuttard’s Engines are Not Within the Shader Core

Petitioner identified Stuttard’s plurality of engines as the channel controller 108, array controller 104, thread manager components, and binning unit 106. Pet., 55. However, Petitioner never relied on the binning unit and withdraws its identification of the binning unit as an engine. AMD does not dispute that the remaining engines satisfy the claims.

3. Stuttard’s EPU 8 is a Scheduling Module as Claimed

The “scheduling module” in Stuttard—EPU 8—is outside both the host system (second processing unit) and processing core 10 (first processing unit). As explained *supra* Section IV.A.2., the scheduling module does not need to be “within” the second processing unit to be “associated” with it. Under the prosecution disclaimer, the “scheduling module associated with a second processing unit” *cannot be* within the *first* processing unit, it does not require the scheduling module to be within the second processing unit. *Id.* The “scheduling module” can be outside both the first and second processing units. *Id.*

4. The Claims Do Not Require “Parallel” Engines or that Each Engine Perform Every Claim Function

AMD’s arguments at POR, 74-75 are based entirely on its improper construction that the claims require “parallel” or “independent” engines. *See supra* Section II.B.

B. Claim 19

AMD argues that Stuttard’s “general system control” task is not a “general-compute” task. The ITC found it “persuasive” that Stuttard’s disclosed “general system control” task would be understood as a type of “general-compute task.” Ex. 2003, 178-179. *See* Pet. 61-62.

Dr. Pedram testified that a “general-compute task” is “something other than a graphics processing task.” Ex. 2007, 113:17-25. Dr. Hart agreed, testifying that “A POSITA would have understood that a non-graphics task consisted of a plurality of what were known as compute shader threads,” and a compute shader was “an example of a general compute task.” Ex. 1009, 128:13-129:21; *see also* 137:3-9.

AMD does not dispute that a general system control task is a non-graphics task, which both experts agree is a “general-compute task.” AMD’s only remaining argument is that a “general system control” task does not perform “computations.” Nonsense. Stuttard discloses that “each thread may be assigned to any one of a number of tasks” and “thread zero may be assigned for general system

control.” Ex. 1006, 11:16-25. Stuttard discloses that “instruction streams for the processing blocks 106 are known as ‘threads.’” *Id.*, 10:9-10. A POSITA would recognize (if it is not inherent) that an instruction stream, executed on a processor block, will perform computations. Ex. 1012 ¶46. And while not relevant, AMD omits Dr. Pedram’s testimony that I/O actions can support or be “part of a computational task.” Ex. 1007, 148:15-150:19, 151:1-6.

C. Claim 20

AMD admits there is a “thread priority” in Stuttard—its argument as to claim 20 is based on its wrong construction of “task.” Based on Stuttard’s teaching that “[t]he threads are assigned priorities relative to one another,” and properly finding threads to correspond with the claimed “tasks,” the ITC found it “convincing” that threads “will be passed to the processing blocks 106 based on that priority,” and that “supporting details of this priority-based decision making are disclosed elsewhere in the reference.” Ex. 2003, 180; Ex. 1006, 10:5-12:22, 23:3-26:10. Because the thread manager and scheduler use priorities to determine when to issue or schedule threads, it is at least obvious that threads (tasks) are received by the processing blocks based on their priority. *Id.*; Pet. 62-63. *See* Ex. 2007, 152:7-14 (testifying that the threads resulting from the decomposition of a high priority task will *all* be high priority, regardless of whether they have the same priority level as a parent task).

AMD's other argument is that when a thread pauses, the thread executed next may depend on whether it is "yielding." This is irrelevant. Nonetheless, Stuttard states that high priority threads should not be treated as yielding, and even during a yield event, the schedule activates the highest priority thread that is ready. Ex. 1006, 24:30-32, 25:5-10.

Dated: April 10, 2024

Respectfully submitted,

/Jeffrey L. Johnson/

Jeffrey L. Johnson
Reg. No. 53,078
BAKER BOTTS, LLP
910 Louisiana Street
Houston, TX 77002-4995
Telephone: 713-229-1234
Facsimile: 713-229-1522
Email: Jeffrey.Johnson@bakerbotts.com
DLBBRealTEKIPRs@bakerbotts.com

Christopher J. Higgins
Reg. No. 66,422
ORRICK, HERRINGTON & SUTCLIFFE LLP
Columbia Center
2100 Pennsylvania Avenue NW
Washington, D.C. 20037
Telephone: (202) 339-8400
Facsimile: (202) 339-8500
Email: 0CHPTABDocket@orrick.com

Steve Baik
Reg. No. 42,281

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

*Counsel for Petitioner Realtek
Semiconductor Corp.*

CERTIFICATE OF SERVICE

In accordance with 37 C.F.R. § 42.6(e), the undersigned certifies that on April 10, 2024, a complete and entire copy of this **PETITIONER'S REPLY TO PATENT OWNER'S RESPONSE TO PETITION** and Exhibits 1009-1013 were served on Patent Owner via electronic mail at the following addresses:

William A. Meunier
wameunier@mintz.com

Michael T. Renaud
mtrenaud@mintz.com

Adam S. Rizk
arizk@mintz.com

Reza Dokhanchy
rdokhanchy@mintz.com

AMD_IPRS@mintz.com

/Jeffrey L. Johnson/
Jeffrey L. Johnson (Reg. No. 53,078)
Lead Counsel for Petitioner

**CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME
LIMITATION, TYPEFACE REQUIREMENTS, AND TYPE STYLE
REQUIREMENTS**

1. This **Petitioner's Reply to Patent Owner's Response** complies with the type-volume limitation of 5,600 words, comprising 5600 words, excluding the parts exempted by 37 C.F.R. § 42.24(a).

2. This **Petitioner's Reply to Patent Owner's Response** complies with the general format requirements of 37 C.F.R. § 42.6(a) and has been prepared using Microsoft® Word in 14-point Times New Roman.

Respectfully submitted,

BAKER BOTTS L.L.P.

/Jeffrey L. Johnson/
Jeffrey L. Johnson (Reg. No. 53,078)
Lead Counsel for Petitioner

Date: April 10, 2024