

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

REALTEK SEMICONDUCTOR CORP.,
Petitioner,

v.

ATI TECHNOLOGIES ULC
Patent Owner.

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

PATENT OWNER'S SUR-REPLY TO PETITIONER'S REPLY

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. CLAIM CONSTRUCTION	1
A. “task”	1
III. WILT	3
A. All Claims.....	3
1. Wilt Does Not Disclose A Shader Core That Is Configured To Simultaneously Execute Two Tasks	3
2. Dr. Pedram Opined The Engines Of The Multi-PPU Theory Are Non-Sensical	5
3. Petitioner’s Alleged Shader Core Is The PPU (Or Collection Of PPU’s), Not the Cores Within The PPU’s, And Thus The Alleged Engines Are Within The Alleged Shader Core.....	6
B. Claim 19	7
1. Multi-PPU Theory Does Not Execute Graphics- Processing Tasks While Executing General-Compute Tasks	7
2. Single-PPU Theory Is Waived.....	8
C. Claim 20	9
IV. JIAO.....	10
A. All Claims.....	10
1. The Alleged Engines Are Not The Claimed Engines	10
a. Jiao’s Shaders Do Not Receive Tasks, Much Less The Tasks Executed By The Engines	10
b. Jiao’s Thread Slots Are Part Of The Shader Core, Else The Shader Core Does Not Receive Tasks.....	12
c. Jiao’s Shaders Are Part Of The Shader Core	13
2. The Alleged Scheduling Module Is Not Within The Second Processing Unit	14

3.	The Tasks Generated By The Shaders Are Not The Claimed Tasks.....	14
4.	Petitioner Has Not Shown The Shader Core Simultaneously Executes Multiple Tasks	15
B.	Claim 19	16
C.	Claim 20	19
V.	STUTTARD	20
A.	All Claims.....	20
1.	No Disclosure Of Receiving And Simultaneously Executing Tasks (Or Threads)	20
2.	Petitioner Waived Any Argument Its Engines Are Not Part Of The Shader Core.....	22
3.	EPU 8 Is Not A Scheduling Module.....	23
B.	Claim 19	23
C.	Claim 20	24
VI.	CONCLUSION.....	25

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>Ariosa Diagnostics v. Verinata Health, Inc.</i> , 805 F.3d 1359 (Fed. Cir. 2015)	11
<i>Bilstad v. Wakalopulos</i> , 386 F.3d 1116 (Fed. Cir. 2004)	2
<i>DSS Tech. Mgmt., Inc. v. Apple Inc.</i> , 885 F.3d 1367 (Fed. Cir. 2018)	24, 25
<i>Henny Penny Corp. v. Frymaster LLC</i> , 938 F.3d 1324 (Fed. Cir. 2019)	9
<i>Intelligent Bio-Systems, Inc. v. Illumina Cambridge, Ltd.</i> , 821 F.3d 1359 (Fed. Cir. 2016)	1, 4, 17
<i>Wasica Fin. GmbH v. Cont'l Auto. Systems</i> , 853 F.3d 1272 (Fed. Cir. 2017)	<i>passim</i>

TABLE OF EXHIBITS

Exhibit	Description
2001	Declaration of Dr. John Hart and CV
2002	Pat Hanrahan and Jim Lawson. 1990 “A Language for Shading and Lighting Calculations.” Proc. SIGGRAPH ’90. pp. 289-298
2003	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318, Final Initial Determination (July 7, 2023)
2004	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318, Respondents Post-Hearing Brief (April 10, 2023) (Redacted)
2005	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318, Deposition Transcript of Dr. Pedram (January 4, 2023)
2006	Declaration of John Hart Regarding U.S. Patent No. 8,854,381
2007	Deposition of Massoud Pedram (January 5 and 8, 2024)
2009	Declaration of Dr. John Hart Regarding U.S. Patent No. 8,854,381 (May 22, 2024)

I. INTRODUCTION

Petitioner’s Reply only confirms its Petition fails. Upon careful analysis as explained below, each ground has multiple critical flaws. Faced with these deficiencies, Petitioner raises at least *thirteen* new arguments asserting obviousness, inherency, and theories relying on previously unidentified portions of prior art, which fail both on the merits and procedurally. The Petition could have, and should have, included these arguments. As the Federal Circuit provides, “[i]t is of the utmost importance that petitioners in the IPR proceedings adhere to the requirement that the initial petition identify ‘with particularity’ the ‘evidence that supports the grounds for the challenge to each claim.’” *Intelligent Bio-Systems, Inc. v. Illumina Cambridge, Ltd.*, 821 F.3d 1359, 1369 (Fed. Cir. 2016). Even if the Board considers these untimely arguments, they still fail to establish unpatentability of the challenged claims.

II. CLAIM CONSTRUCTION

A. “task”

Petitioner provides Dr. Hart opined at the ITC that a task comprises “one or more” threads. Reply, 1. But, consistent with his opinion here, Dr. Hart also previously opined “a ‘task’ includes a ‘plurality of ... threads.’” Ex. 1010, ¶78 (citing Ex. 1001, 1:47-2:7). Petitioner further misconstrues Dr. Hart’s testimony with respect to shader core’s receipt of tasks. Reply, 1-2. Dr. Hart did not say a

shader core must recognize tasks to “receive tasks,” nor did he apply that requirement in his analysis. Ex. 1009, 174:3-179:7. Rather, he gave one example of when a shader core has received the tasks that were received by the engines: if the shader core receives state data associated with a task, then a POSITA would know the threads being executed by the shader core are associated with that task. *Id.*, 73:5-10, 117:14-19. Dr. Hart explained that when a shader core executes threads, it’s the prior art’s lack of correspondence of those threads to distinct tasks he takes issue with. *Id.*, 174:3-175:14, 181:2-7, 171:9-15.

Petitioner incorrectly argues the ’381 patent allows a “task” to be “one or more” threads. Reply, 2. The ’381 patent doesn’t state a “task” includes “one or more” threads. Instead, it explicitly states a task includes “a plurality of threads.” Ex. 1001, 1:47-2:7. The term “plurality” means more than one. *Bilstad v. Wakalopoulos*, 386 F.3d 1116, 1122 (Fed. Cir. 2004) (“plurality” construed as range “bounded by two and infinity”). Further, the specification only refers to a “task” as having “threads.” Ex. 1001, 1:58-59, 2:23-38, 6:45-47; POR, 22-23. Therefore, the patentee intended a “task” to include at least two threads.

Petitioner’s claim that Dr. Hart offered no opinions under the ITC’s construction of “task” is inaccurate. His opinions apply regardless of the construction of “task,” and was offered to clarify a shader core executing two threads doesn’t necessarily disclose that it is executing two tasks, something Dr. Pedram

agrees with. As Dr. Pedram testified, “*the task and the thread are not the same construct*,” because threads are “executed in the end on [a] piece of hardware,” whereas tasks have to be “[t]ransformed in such a way that it can . . . get mapped into hardware that implements it.” Ex. 2007, 21:21-22:24.

Lastly, Dr. Pedram first repeatedly contended a task is not a thread and vice versa. Pet., 41; Ex. 1003, ¶102. Both experts agree on this. Only Petitioner differs.

III. WILT

A. All Claims

1. Wilt Does Not Disclose A Shader Core That Is Configured To Simultaneously Execute Two Tasks

The claims require the shader core is “configured to . . . execute a first task from the plurality of tasks while executing a second task from the plurality of tasks.” *E.g.*, Ex. 1001, claim 1. Although Wilt states its cores operate on tasks, those “tasks” are at best commands, *e.g.*, instructions, and therefore not the “tasks” claimed in the ’381 patent. POR, 41-43. Instead, Wilt’s cores operate two levels of abstraction lower. *Id.* The ’381 patent’s “task” comprises multiple threads, and each thread comprises multiple instructions (*i.e.*, multiple commands). Thus, even if Wilt’s cores simultaneously execute those commands, there is no evidence that those commands were decomposed from different tasks, and thus no evidence the cores simultaneously execute different tasks. POR, 43-45. This is true of both Petitioner’s Single-PPU Theory and Multi-PPU Theory. *Id.*

Petitioner claims “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.” Reply, 4. That is incorrect. Patent Owner’s argument is the above: there is no proof what is executed, allegedly simultaneously, was decomposed from, different tasks.

Petitioner’s Single-PPU Theory argument that the “divergent execution paths” of SIMT means each core executes two or more different instruction streams corresponding to different tasks simultaneously is incorrect. The “SI” of SIMT stands for “single instruction.” Ex. 1003, ¶75. SIMT does not mean two different instructions are being executed simultaneously—by definition it must be the same *single* instruction. Ex. 2009, ¶3.

Petitioner newly asserts it is inherent or obvious that different tasks may be executed simultaneously. Reply, 6. This argument was never asserted in the Petition and is waived. Pet., 28; *Intelligent Bio-Systems*, 821 F.3d at 1369-70 (affirming disapproval of a party’s “entirely new rationale” raised in the reply). Further, Petitioner provides these assertions without any underlying support.

Petitioner also points to Wilt’s ability to execute a “wide variety of applications” as proof of simultaneous execution of different tasks. Reply, 6 (citing Ex. 1004, 4:33-40). All this suggests is that Wilt has the ability to execute tasks

from different applications, not that Wilt is executing any tasks simultaneously. Ex. 2009, ¶4.

2. Dr. Pedram Opined The Engines Of The Multi-PPU Theory Are Non-Sensical

Petitioner cites caselaw providing separate components can be mapped to single claimed unit. Reply, 7. But this generalization flies in the face of Petitioner’s *expert’s concession* that the front-end unit, work distribution unit, and core interface should not be considered together as one unit. POR, 45-46; Ex. 2007, 69:18-70:3, 71:13-72:1, 72:2-73:9 (“a person skilled in the art would not even try to establish a correspondence between . . . the combination of these three units, and the engine”).

Petitioner’s non-precedential case is irrelevant. Reply, 7 (citing *Textron Innovations, Inc. v. American Eurocopter Corp.*, 498 Fed. Appx. 23 (Fed. Cir. 2012)). The term there was “replacement” in the preamble, in the phrase “[a]n improved replacement helicopter landing gear assembly . . .”; the court held “it is apparent that the term ‘replacement’ is *intended to describe the principal intended use of the invention*, but not to import a structural limitation . . .” *Id.*, 24, 29. By contrast, this case is more like the cases the Court distinguished because the term “engines” is not in the preamble and provides structural limitations. *Id.*, 28. As Dr. Pedram conceded, a POSITA would not consider the random assortment of items

Petitioner points to as “an” anything, let alone an engine. There are limits to box-drawing, and Petitioner exceeds those limits.

3. Petitioner’s Alleged Shader Core Is The PPU (Or Collection Of PPUs), Not the Cores Within The PPUs, And Thus The Alleged Engines Are Within The Alleged Shader Core

The Petition unambiguously and repeatedly states Petitioner’s two theories, naming a PPU or collection of PPUs as the “shader core:” “Each parallel processing unit (PPU) 202 *is the claimed ‘shader core* configured to receive the plurality of tasks,”” and “[a]lternatively, the plurality of parallel processing units (PPUs) 202 *is the claimed ‘shader core’.*” Pet., 28, 29 (“ . . . a PPU 202 (*which is the claimed ‘shader core’*). . .”); POR, 29-31.

Petitioner identifies the Petition’s statement “the processing cores 208” as the claimed ‘processing elements’ of the ’381 Patent” as allegedly supporting a new theory that something other than the PPU is the shader core. Reply, 9. That only proves the point. Petitioner’s allegation was to satisfy the “processing elements” of claim 16, not the “shader core” of claim 15 as claim 15 doesn’t require “processing elements.” Pet., 29.

Petitioner may not now change its theory so that some unstated sub-component of a PPU (or of multi-PPUs) is now alleged to be the claimed shader core. As a result, the alleged engines (work distribution, front-end, and core interface units) are improperly within the alleged shader core. POR, 47-51.

B. Claim 19

1. Multi-PPU Theory Does Not Execute Graphics-Processing Tasks While Executing General-Compute Tasks

Wilt discloses two mutually exclusive embodiments, one for graphics and one for general purpose: “*In one embodiment*, the parallel processing subsystem 112 incorporates circuitry optimized for graphics and video processing *In another embodiment*, the parallel processing subsystem 112 incorporates circuitry optimized for general purpose processing” Ex. 1004, 3:24-31. There is no disclosure Wilt’s system can simultaneously execute both task types. POR, 30-32.

Further, nowhere did Petitioner attempt to prove Wilt can process graphics-processing tasks *while* processing general-compute tasks. Nor has Petitioner identified any explicit or implicit disclosure of Wilt providing so. To disclose such, a more comprehensive description is required, and a POSITA would not find Wilt invented that type of system given Wilt’s vague and non-enabling statements. Ex. 2009, ¶5.

Even if Petitioner were correct (Patent Owner disagrees), Wilt still only discloses the mere capability to perform both types of tasks, but not actual configuration, as required. Petitioner must show, *e.g.*, disclosures in which engines are receive both types of tasks, and send to the shader core both types of tasks for processing simultaneously. Petitioner did not. Ex. 2009, ¶6.

Petitioner argues for the first time Wilt's PPUs "may be operated in parallel to process data at a higher throughput than is possible with a single PPU." Reply, 10. This says nothing about graphics-processing tasks and general-compute tasks being executed simultaneously. Ex. 2009, ¶7.

Lastly, Petitioner argues "it was at least obvious to configure Wilt's system" in the manner of claim 19. Reply, 10. But because Petitioner never raised an obviousness argument in their Petition for this claim, it is waived. Pet., 35. *Wasica Fin. GmbH v. Cont'l Auto. Systems*, 853 F.3d 1272, 1285-87 (Fed. Cir. 2017) (affirming Board's ruling that an obviousness challenge was "insufficiently precise and underdeveloped" because the petitioner "did not make out its obviousness case in its petition"). Further, Petitioner provides no support. The Petition only asserts it is obvious to execute two tasks simultaneously, but not specifically graphics-processing tasks and general-compute tasks.

2. Single-PPU Theory Is Waived

Petitioner claims it asserted the Single-PPU theory for claim 19. Reply, 10-11. However, *nowhere* did the Petition allege a *single* PPU can process both types of tasks required by claim 19, either directly or through obviousness. Pet., 35; POR, 29-30. The cited-to pages in footnote 13 of the Reply apply only to the execution of different alleged "tasks" without discussing the type of task (setting aside these so-called "tasks" are truly instructions, as explained above). This argument is waived.

Henny Penny Corp. v. Frymaster LLC, 938 F.3d 1324, 1330-31 (Fed. Cir. 2019) (affirming Board’s refusal to permit new reply argument). Further, Petitioner identifies no support for its waived assertions, nor does Dr. Pedram. Ex. 1012, ¶41.

C. Claim 20

Wilt does not disclose any priority. POR, 36-37. Petitioner asserts Wilt’s context-swapping *necessarily* involves task priorities. Petitioner claims each task “may” require “a different PPU context,” a new argument not made in the Petition, and thus waived. Reply, 11. Even if true, implicit behind Petitioner’s claim is each task doesn’t always have to have a different PPU context.

Context-swapping is used for several reasons unrelated to priorities associated with tasks, such as when a thread has stalled execution and a new thread is swapped in. POR, 37; Ex. 2006, ¶¶90-91. Dr. Pedram agrees there are several reasons to do context-swapping, *e.g.*, when data is not available yet (*i.e.*, a stall):

But there could be some other use -- situations in which you do context swap. For example, if a thread is running and gets to a point in its execution that it needs certain data item that’s not available, then instead of waiting for the data item to become available and therefore keep the processing unit in standby state doing nothing, you context swap it. Bring another task in to run to maximize utilization of the processing element itself. So that’s another common scenario.

Ex. 2007, 60:20-61:4. Nowhere does this concern priority. Thus, context-swapping doesn’t imply it is used to enable a prioritization scheme.

Petitioner identifies Patent Owner's infringement allegations in the 177 proceeding, *i.e.*, the Patched Mali Driver orders jobs based on their contexts in a "head" to "tail" list. Ex. 1013, 70. That the assignment of priorities in the accused products is based on the data in contexts is irrelevant to the question here, which is does mere disclosure of context-swapping disclose priorities. As explained, the answer is "no."

IV. JIAO

A. All Claims

1. The Alleged Engines Are Not The Claimed Engines

a. Jiao's Shaders Do Not Receive Tasks, Much Less The Tasks Executed By The Engines

The claims require tasks are received by engines, and then received and executed by the shader core. Ex. 1001, claim 15. Petitioner has not presented any construction that changes this plain meaning.

First, Petitioner argues the "commands" generated by driver software in Jiao are the claimed tasks and that they are received by Jiao's shaders. Reply, 12. But Jiao only states the commands are communicated to "registers in the GPU 114." Ex. 1005, 5:17-23. Nothing suggests these specific registers are within the shaders, nor has Petitioner attempted to prove such. Petitioner points to Ex. 1005, 5:47-51, 9:41-49. Reply, 12. The first citation states the computational core 204 is capable of meeting the computational requirements imposed by the shader tasks. The second

disclosure states EU pool control unit assigns the tasks generated by the shaders to available threads of the execution units. None of these disclosures provide the shaders *receive* anything other than data. POR, 52-55; Ex. 2009, ¶8. If the shaders received tasks, Jiao would say so, just like it explains in great detail how the shaders receive data.

Petitioner relies on the Board’s finding the shaders perform operations “responsive to a command.” Reply, 13. But Petitioner ignores the latter, contradictory, part of Jiao: “*Responsive to a command* from the driver software 128, such as to draw a triangle, *vertex information is passed to vertex shader 304* in the computational core 204 to implement vertex transformations.” Ex. 1005, 7:57-60. Thus, Jiao provides, *responsive to a command*, only data is passed to the vertex shader, and the shader performs pre-programmed tasks in response to receiving the data.

Petitioner newly relies on the ’381 patent to support finding what is generated by Jiao’s driver must be received by Jiao’s shaders. Reply, 13 (asserting the ’381 patent allegedly “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”). This is another waived argument. *Ariosia Diagnostics v. Verinata Health, Inc.*, 805 F.3d 1359, 1367 (Fed. Cir. 2015); Pet., 39-

47. Further, the only thing the '381 patent provides is the driver translates “standard code . . . into a native format of instructions understood by the GPU.” Ex. 1001, 1:41-44. There is no disclosure of tasks or engines, especially engines that “provide tasks that are mapped to the threads of the execution units” as claimed by Petitioner. Ex. 2009, ¶9. Given Jiao’s explicit disclosures of how its shaders only receive data, it is incorrect to conclude Jiao’s alleged shaders received tasks, and that the alleged shader core receives those tasks.

**b. Jiao’s Thread Slots Are Part Of The Shader Core,
Else The Shader Core Does Not Receive Tasks**

Petitioner wrongly claims the Petition made clear Jiao’s thread slots are carved-out from the alleged shader core. Pet., 40 (providing alternative theory includes “the ‘thread slots’ *within execution units 420 of the EU pool 414*,” the alleged shader core). Jiao provides, and Petitioner acknowledges, the thread slots are part of the alleged shader core, EU Pool. Ex. 1005, 9:18-21; POR, 55-56.

The claims require “task” be received twice: (i) by the engines; (ii) by the shader core. Petitioner’s “thread slot” theory only has one such alleged reception: by the thread slots, the “engines.” Pet., 40. Even if Petitioner was allowed to carve-out the thread slots from the alleged shader core, Petitioner provides no evidence of reception of tasks by the shader core. If the thread slots aren’t carved-out from the

shader core, then part of the alleged engines (the thread slots) is within the alleged shader core. Neither theory satisfies the claim requirements.

c. Jiao's Shaders Are Part Of The Shader Core

Jiao's shaders are executed on EUP 414, the alleged shader core. Ex. 1005, 9:25-31 (“[G]raphics processor 202 utilizes programmable vertex 304, geometry 306, and pixel shaders 310. Rather than implementing the functions or operations of these components as separate fixed function units . . . , *the operations are instead executed by the pool 414* of execution units.”); POR, 56. The Reply argues because the shaders send “tasks” to the execution units, they cannot be part of the alleged shader core. But the execution units are not the alleged shader core. The larger component, EUP, is. Pet., 47-48 (“A [POSITA] would have understood Jiao’s EU pool (EUP) 414 to be a ‘shader core’ as recited in claim 15”), 48 (“Jiao discloses that the EU pool (EUP) 414 (which is the claimed “shader core”) comprises a plurality of EUs 420.”). Thus, Jiao’s shaders (within EUP) can send alleged “tasks” to EUs 420 within EUP, as alleged by Petitioner, and are also within the alleged shader core (EUP), as Patent Owner asserts.

The Petition alleged the larger component (EUP 414) satisfied claim 15’s “shader core,” and alleged the sub-components of EUP 414 (EUs 420) satisfy claim 16’s requirement for sub-components of the shader core, the claimed first and second plurality of processing elements. Petitioner cannot rewrite its allegations now so

that the sub-components (EUs 420) are now the shader core to avoid the problem it created by drawing the shader core box so large it encompasses the alleged engines.

2. The Alleged Scheduling Module Is Not Within The Second Processing Unit

Petitioner newly argues the alleged scheduling module may be outside the alleged second processing unit because the '381 patent provides an example of an OS scheduler that resides in “system memory” and not within the CPU. Reply, 14-15. That is incorrect. First, the '381 patent doesn't state the OS scheduler resides in system memory. *E.g.*, Ex. 1001, 6:42-50, 10:22-32. Further, even if it did reside in system memory as code, which appears to be Petitioner's argument, the OS scheduler cannot run merely sitting in memory. Ex. 2009, ¶10. For the scheduling module to perform its claimed functions, it must be executing on the CPU, meaning it is within the claimed “second processing unit.” *Id.* The fact that Fig. 4 of the '381 patent depicts the OS schedule as “outside” of CPU doesn't negate this. For example, a POSITA would readily recognize applications are also executed on the CPU, but they are also drawn as being “outside.” Lastly, the system memory is part of the CPU. *See also* POR, 58-59.

3. The Tasks Generated By The Shaders Are Not The Claimed Tasks

Petitioner incorrectly claims Patent Owner's arguments are premised on the wrong construction of “task.” Reply, 16. The issue is not how many threads a task

is, but simply the alleged tasks are generated in the first instance by the shaders. POR, 59-61. They have no relationship to what Petitioner alleges the shaders receive, which are merely “commands,” without data, whereas the alleged “tasks” created by the shaders are generated by operating on the data. So the question is not whether tasks can be modified, but why there is any reason to believe the tasks generated in the first instance by the shaders of Jiao are even a modified version of what they receive. Moreover, the ’381 patent’s preprocessing logic doesn’t modify the tasks, nor does the Reply identify any modification. POR, 59-60. Petitioner fails to address the actual disclosures in the ’381 patent, or Patent Owner’s explanation thereof. Reply, 16.

4. Petitioner Has Not Shown The Shader Core Simultaneously Executes Multiple Tasks

Petitioner again misstates Patent Owner’s argument regarding the claims’ requirement of simultaneously executing two distinct tasks. It is not that executing a thread is not executing a task. It is that Jiao (like all references asserted) doesn’t disclose any configuration in which simultaneously executing threads are decomposed from different tasks. Moreover, even if Patent Owner’s construction is wrong and a thread may be a task, Petitioner has not shown Jiao’s alleged engines send the alleged “tasks” to the shader core in a manner which would allow the shader core to execute them simultaneously (*e.g.*, sending them at the same time or in an

overlapping manner). Mere happenstance is insufficient to invalidate claims reciting a system “configured to” perform particular actions. POR, 25-27.

Petitioner claims it is incorrect multiple of Jiao’s “tasks” have a single context. This is correct based on the disclosures provided in Dr. Hart’s declaration. Ex. 2006, ¶116 (citing Ex. 1005, 13:52-53, 14:29-31, 14:34-36). For example, Jiao provides “[w]hen the counter 904 reaches zero, the EU pool control logic 605 is notified that *all tasks for the context* have completed and the EU pool 414 is flushed.” Ex. 1005, 14:34-36 (emphasis added). Petitioner did not address any of these disclosures. Reply, 16-17. Further, Petitioner’s cited portions of Jiao doesn’t support finding “concurrently running tasks (‘shader programs’) may be associated with multiple rendering contexts in each execution unit.” Ex. 2009, ¶11. The quote by Petitioner doesn’t support that the EUs are “simultaneously running multiple contexts.” Ex. 1005, 10:38-39. Instead, this is a reference to the access paths, and provides they “include operations involving the constant buffer 510 and context address translation table(s) 512 . . . simultaneously run[] multiple contexts.” *Id.* This means the buffer which stores the contexts can store multiple contexts at the same time. Ex. 2009, ¶11.

B. Claim 19

Petitioner’s reliance on the ITC’s “more than likely” finding does not resolve Jiao’s deficiencies. Reply, 17. First, Petitioner wholly misrepresents the ITC

decision: “While it seems more than likely . . . it was known that *GPUs* could be used for general compute tasks, . . . Respondents have not made a clear and convincing case that the EUs 420 of Jiao already function in this way or could be made to with modification.” Ex. 2003, 168. Thus, the ITC did not find it “more than likely” *Jiao specifically* could be used for general-compute tasks, only that it was “more than likely” *GPUs generally* could be. Second, Petitioner did not articulate this argument in its Petition, thus it is waived. *Intelligent Bio-Systems*, 821 F.3d at 1369-70.

Regardless, the claims require the two different task-types be executed simultaneously. Jiao has no such disclosure; the shaders are graphics-specific components. POR, 32-34. It is incorrect that the functionality of instruction decoding, operand fetch, branch computation, instruction execution, shift/load, load/store relates to general-compute tasks specifically. Ex. 2009, ¶12. Rather, these are generic actions pertaining to all instructions, as provided by some of the names, *e.g.*, instruction decoding, instruction execution. *Id.* The other functionality relates to how instructions are read so their missives can be executed by the processor. *Id.* None of this functionality is so specific to general-compute tasks that would enable a POSITA to conclude Jiao can process general-compute tasks. *Id.*

Nor does Jiao’s “programmable” shaders enable general-compute tasks. Petitioner claims Jiao’s pixel, vertex, and geometry shaders are mere “examples.”

Reply, 18. This is a gross overstatement; what Jiao actually states is: “various shader programs” or “various types of shader programs” include[e a] vertex shader, geometry shader, and/or pixel shader.” Ex. 1005, 5:49-51, 6:59-61; Ex. 2009, ¶13. These phrases do not support finding Jiao’s shaders are examples which can be programmed to execute general-compute tasks. Therefore, it cannot be concluded that Jiao’s “programmable” shaders disclose general-compute tasks.

Petitioner takes issue with Dr. Hart not defining “general-compute task.” But he was not required to. Dr. Hart provided examples instead, *e.g.*, general-compute tasks “refer to compute shader threads and do not utilize the other graphic-specific machinery of a processor.” Ex. 1009, 130:2-14, 133:2-18. Further, the fact that graphics-processing tasks may include computations doesn’t mean they are also general-compute tasks. Reply, 18. This is belied by Petitioner’s later (incorrect) argument that *any non*-graphics-processing task is a general-compute task. Reply, 24. If both Petitioner’s positions were true, the term “general-compute” would be meaningless because both graphics-processing tasks and non-graphics-processing tasks would all be “general-compute” tasks; in fact, *all* tasks would be “general-compute” tasks.

Dr. Pedram confirmed Jiao doesn’t state EUDP is in the alleged shader core, EUP 414. *See* Ex. 2007, 119:3-120:11. Nor does Jiao at 10:33-39 (which Petitioner cites), which only provides that fast access path 506 is implemented in

computational core (not EUP 414), *via* EUDP, meaning EUDP may be connected to the computational core in a manner similar to the shader core receives tasks from the engines, as provided by claim 15. Ex. 2009, ¶14. Furthermore, Petitioner’s late assertion that it would be obvious that EUDP is within the alleged shader core is waived. Reply, 19 (citing Ex. 2007, 119:6-19); *Wasica*, 853 F.3d at 1285-1287.

C. Claim 20

Petitioner argues that constants with priorities impact the receipt of tasks by the shader core. But this focuses on only part of the limitation, *i.e.*, “the shader core is configured to receive . . . tasks.” It overlooks the remainder requiring claimed priorities to be “associated with each of the plurality of tasks.” Thus, Petitioner has not shown this claim is disclosed or obvious because there is no disclosure in Jiao of task-associated priority. POR, 38.

Further, Petitioner asserts a task is not sent to the execution unit until its rendering context is loaded. Reply, 19. Petitioner did not make this argument in its Petition, and it is waived. Regardless, even if true, all this means is loading the rendering context is a prerequisite for the task being sent to the shader core; there is no task-associated priority disclosed. Ex. 2009, ¶15.

V. STUTTARD

A. All Claims

1. No Disclosure Of Receiving And Simultaneously Executing Tasks (Or Threads)

Petitioner incorrectly states Patent Owner's arguments are based on an improper construction of "task." Stuttard cannot execute multiple threads simultaneously, yet alone tasks. POR, 69-70. Petitioner never attempted to rebut Stuttard's disclosures that it is a single SIMD, *id.*, a substantial deficiency. Reply, 20-23

Petitioner first asserts Stuttard's "engines" and "shader core" receive "tasks." Reply, 20-21. But Petitioner acknowledges "instructions provided by EPU 8" are received first by the "engines," and only *after* that by the "shader core." *Id.* Even if the instruction streams were tasks, the shader core isn't configured to execute two tasks (or instruction streams) simultaneously. POR, 66-72.

Petitioner next incorrectly argues Patent Owner didn't address the division of processing elements (PEs) into processing blocks (PBs). *See* POR, 70-71 (reasons for dividing PEs into PBs include operating on different regions of the display or different data groups, redundancy, etc.). Stuttard explains its structure clearly and why it divides PEs into PBs:

A data processing apparatus comprises *a SIMD* (single instruction multiple data) array (10) of processing elements. *The processing elements are operably divided into a plurality of processing blocks, the*

processing blocks being operable to process respective groups of data items.

Ex. 1006, Abstract. Further, after binning, each PB “operates on data associated ***with a single region of the display.***” *Id.*, 22:4-10. Petitioner fails to address the binning units, claiming operating on different regions of the display and redundancy has “little practical benefit.” Reply, 20-23; Pet., 59-60. Because the binning units are crucial to Stuttard’s operation, Ex. 1006, 20:23-25, Fig. 9, this is far from providing “little practical benefit” — it is the purpose of Stuttard’s architecture.

Petitioner argues “Stuttard explicitly discloses that different blocks may be assigned to different tasks.” The only “proof” Petitioner (and Dr. Pedram) identify is what the ***thread manager stores***, which says nothing about the PEs/PBs. Reply, 22 (citing Ex. 1006, 11:19-25); Ex. 1012, ¶37. This single disclosure contrasts with the several disclosures providing Stuttard only executes one thread/instruction at a time. POR, 69-70 (citing Ex.1006, 2:25-34, 12:15-18, 23:18-21, 23:34). While Petitioner claims these are “particular embodiments,” Petitioner hasn’t identified any contrary embodiments.

Further, the use of multiple thread processors doesn’t support simultaneous execution of multiple threads/tasks. POR, 69-70. Thread processors do not decide which instructions are being executed by the PEs/PBs (that is the job of the thread scheduler). Ex. 1006, Fig. 4, 12:16-18. Rather, thread processors “allow for high

level flow control to be performed for a thread,” and assemble instructions to be issued to array controller 104 and channel controller 108 (both of which there is only one). Ex. 1006, 26:3-7, Fig. 3. Having multiple thread processors enables Stuttard to time-interleave instructions from different threads, avoiding stalls. Further, storing multiple threads is beneficial even in a SIMD system because the system can switch threads if the active thread stalls. *Id.*, 23:35-25:31. This all prevents downtime.

If Petitioner asserts one thread can be decomposed from two different tasks, this is incorrect. Reply, 22. Stuttard’s statement “[e]ach thread works co-operatively with other threads to perform a task or tasks” means the threads *altogether* work together to perform a task or multiple tasks, not that a single thread can perform multiple tasks. Ex. 1006, 10:10-12; Ex. 2009, ¶16.

Lastly, Petitioner newly claims “[a]lternatively, a POSITA would understand such configuration to be obvious.” Reply, 22-23. Obviousness for this element was not raised in the Petition. Pet., 55-59. Therefore, it is waived. *Wasica*, 853 F.3d at 1285-1287.

2. Petitioner Waived Any Argument Its Engines Are Not Part Of The Shader Core

Petitioner alleged: “*The binning unit 1069*, the channel controller 108, the array controller 104, and components of the thread manager 102 (including the cache

memory unit 1024, the thread processors 1026, and the status block 1030) constitute the claimed ‘plurality of engines.’” Pet., 56. Petitioner now withdraws the binning unit as an engine. Reply, 23. Because this theory wasn’t in the Petition, it is waived. *Wasica*, 853 F.3d at 1285-1287; *see also* POR, 74-75.

3. EPU 8 Is Not A Scheduling Module

The EPU is not a scheduling module because the scheduling module needs to be *within* the “second processing unit” in order to be “associated with” with it. POR, 73-74.

B. Claim 19

Petitioner incorrectly asserts any *non*-graphics-processing task is a general-compute task. Reply, 24. This argument wasn’t in its Petition, and is waived. *Wasica*, 853 F.3d at 1285-1287. Petitioner misconstrues Dr. Hart’s testimony; he did not agree with Petitioner’s assertion. Reply, 24; Ex. 2009, ¶17.

Petitioner’s argument that all non-graphics-processing tasks are general-compute tasks conflicts with Dr. Pedram’s testimony that Stuttard’s “general system control tasks” are not general-compute tasks unless they specifically involve computations. *See* Ex. 2007, 114:1-12 (testifying general-compute tasks “involve[] computation”), 147:2-148:10 (providing examples of general system control tasks which do not have computations). Petitioner has not shown Stuttard’s general system control tasks involve computations. POR, 34-35.

Petitioner additionally claims a POSITA “would recognize (if not inherent) that an instruction stream, executed on a processor block, will perform computations.” Reply, 25. This is another waived argument. *Wasica*, 853 F.3d at 1285-1287. Even if not waived, Petitioner provides no support besides conclusory testimony from its expert. And Dr. Pedram disagreed. *E.g.*, Ex. 2007, 147:2-148:10 (tasks such as state machines are general system control tasks which do not involve computations). Finally, Petitioner relies on Dr. Pedram’s testimony that certain I/O actions “can support or be ‘part of a computational task.’” Reply, 25. This doesn’t provide I/O actions are general-compute tasks by themselves, only that general-compute tasks may include I/O actions. Ex. 2009, ¶17.

C. Claim 20

Petitioner incorrectly argues ATI’s argument is based on the wrong definition of task. Regardless of the construction of “task,” Stuttard does not disclose or render claim 20 obvious.

Petitioner’s argument “it is at least obvious that threads (tasks) are received by the processing blocks based on their priority” was not in the Petition, and is waived. Reply, 62-63; *Wasica*, 853 F.3d at 1285-87. The Petition’s statement “a POSITA would understand” (Pet., 63), is not an obviousness argument, which requires showing motivation to modify. *DSS Tech. Mgmt., Inc. v. Apple Inc.*, 885 F.3d 1367, 1374 (Fed. Cir. 2018) (“[A] patent can be obvious in light of a single

prior art reference if it would have been obvious to modify that reference to arrive at the patented invention.”). Petitioner provides no such showing. Such hand-waving “cannot be used as a wholesale substitute for reasoned analysis and evidentiary support, especially when dealing with a limitation missing from the prior art references specified.” *Id.*

Stuttard’s thread priority is not related to task priority. POR, 38-39. Dr. Pedram recognizes this because threads and tasks are different constructs. Ex. 2007, 21:21-22:24. Patent Owner explained Stuttard’s use of “yielding” threads means that thread’s priority may change. Petitioner claims this is irrelevant. It is not because Petitioner equates thread priority to task priority. Further, Petitioner wrongly claims Stuttard provides high priority threads should never be treated as yielding. Stuttard only says a thread may not yield if the pause is short, and that states it may activate the “lowest priority thread if no thread is ready.” Ex. 1006, 24:28-29, 25:7-9.

VI. CONCLUSION

For the foregoing reasons, the Board should determine Petitioner has failed to prove claims 15-20 of the ’381 patent unpatentable.

Dated: May 22, 2024

Respectfully submitted,

/s/ William A. Meunier

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

Michael T. Renaud (Reg. No. 44,299)

Adam S. Rizk (Reg. No. 66,867)

MINTZ, LEVIN, COHN, FERRIS,

GLOVSKY AND POPEO, P.C.

One Financial Center

Boston, MA 02111

Telephone: 617-542-6000

Facsimile: 617-542-2241

E-mails: WAMeunier@mintz.com;

MTRenaud@mintz.com

ARizk@mintz.com

Reza Dokhanchy (Reg. No. 62,795)

MINTZ, LEVIN, COHN, FERRIS,

GLOVSKY AND POPEO, P.C.

3580 Carmel Mountain Road, Suite 300

San Diego, CA 92130

Telephone: 858-314-1596

RDokhanchy@mintz.com

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24(d), Patent Owner hereby certifies, in reliance on the word count of the word-processing system (Microsoft Office Word 2010) used to prepare this preliminary response, that the number of words in this paper is 5,599. This word count excludes the tables of contents, tables of authorities, certificate of word count, certificate of service, and table of exhibits.

Dated: May 22, 2024

/s/ William A. Meunier

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

CERTIFICATE OF SERVICE

I certify that a copy of Patent Owner's Sur-Reply to Petitioner's Reply and accompanying Exhibit 2009 are being served by electronic mail on the following counsel of record:

Lead Counsel

Jeffrey 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

Backup Counsel

Christopher J. Higgins
Reg. No. 66,422
ORRICK, HERRINGTON &
SUTCLIFFE, LLP
2100 Pennsylvania Avenue NW
Washington, DC 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

Dated: May 22, 2024

/s/ William Meunier

William Meunier (Reg. No. 41,193)