

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

**DECLARATION OF DR. MASSOUD PEDRAM IN SUPPORT OF
PETITIONER'S REPLY**

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. CLAIM CONSTRUCTION	1
A. “task”	1
B. “engine”	5
C. “configured to”	6
D. “each of the plurality of tasks”	6
III. ALL OF THE CHALLENGED CLAIMS ARE OBVIOUS.....	7
A. All Challenged Claims	7
1. Wilt.....	7
a. Wilt Renders Obvious a Shader Core that Simultaneously Executes Different Tasks [Limitation 15[e]].....	7
b. A Shader Core Comprised of the Plurality of Wilt’s PPUs Discloses and Renders the Challenged Claims Obvious	13
c. Wilt’s Engines are Not Within the Shader Core	14
2. Jiao	14
a. Jiao Discloses and Renders Obvious a “Plurality of Engines” as Claimed.....	14
(1) Jiao’s Shaders Receive Tasks.....	14
(2) Jiao’s Shaders Are Not Part of the Shader Core ..	15
b. Jiao’s Scheduling Module is Not Associated with the First Processing Unit.	16
c. Jiao’s Shader Core Receives the Plurality of Tasks	16
d. Jiao Discloses Receiving and Simultaneously Executing At Least Two Tasks	17
3. Stuttard	18
a. Stuttard Discloses Sending and Simultaneously Executing Tasks.....	18
(1) Stuttard’s Engines and Shader Core Receive and Process Tasks	19

(2) Stuttard’s Simultaneously Executes Different Threads	21
b. Stuttard’s Engines are Not Within the Shader Core.....	23
B. Claim 19	23
1. Wilt.....	23
a. Petitioner’s “Single PPU” Theory Invalidates Claim 19	23
b. Petitioner’s “Multiple PPU” Theory Invalidates Claim 19	24
2. Jiao	25
3. Stuttard	27
C. Claim 20	28
1. Wilt.....	28
2. Jiao	29
3. Stuttard	30
IV. DECLARATION	32

I, Dr. Massoud Pedram, declare as follows:

I. INTRODUCTION

1. This is my Reply Declaration in Support of the Petition (“Reply Declaration”). My Reply Declaration responds to the technical issues raised in Patent Owner Response and Dr. Hart’s associated declaration. Nothing in Patent Owner’s Response has changed any of my opinions that the challenged claims are invalid.

II. CLAIM CONSTRUCTION

A. “task”

2. Dr. Hart significantly alters the meaning of the term task from that used at the ITC, which was intended to cover the plain and ordinary meaning in light of the specification. First, Dr. Hart now requires that a task be “decomposable into two or more related threads.” Second, he has excluded “or portion of an application to be executed” from the ITC’s plain and ordinary construction. I disagree with Dr. Hart’s new construction.

3. It is a true statement that tasks can be decomposed, transformed, or divided into threads. Dr. Hart asserts each task can only be decomposable into two or more related threads. I disagree. As I testified, a POSITA would understand that a single task can be comprised (e.g., decomposed into) of a single thread. Ex. 2007, 11:7-16. It is beyond dispute that tasks can be both simple or complex, and that simple tasks can be comprised of only a single constituent thread. For example,

consider a graphics processing system which executes a basic image resizing task using the nearest-neighbor interpolation method. The process, for example, involves scaling the image up, adjusting the resolution and dimensions of the image. This task can be handled by a single thread as it typically requires basic pixel manipulation operations, such as interpolation, and not intricate graphics rendering techniques or complex mathematical computations that would typically utilize multiple threads. Further, the '381 patent is consistent with that plain and ordinary understanding of a task. There is nothing in the '381 patent disclosure that limits the claimed graphics processing systems to only executing complex tasks, nor does the '381 patent ever expressly define a single task as being decomposable into two or more threads. Those words simply are not there. Instead, the '381 patent uses permissive language that would not have been understood to be limiting or to exclude the full scope of the plain and ordinary meaning. For example, the '381 patent states: "To perform a single graphics processing task, a GPU may execute a plurality of different threads (sequence of instructions). Each thread may comprise a shader program...." Ex. 1001, 1:49-57 (emphasis added). Even if Dr. Hart's construction was adopted, it would have no effect on my opinions. A POSITA would have considered it obvious that the graphics processing systems asserted in the prior art would have included tasks that are decomposable into two or more related threads. Graphics processing tasks include hundreds of tasks, some simple and some complex. As I testified

earlier, the number of decomposable threads is nothing more than the level of complexity of the task. For example, Stuttard discloses 3D activities such as calculating vertices, primitives or rastering (tasks) and multiple threads are assigned to executing such tasks. Ex. 1006, 11:21-25. A POSITA would have understood 3D tasks to include tasks that are decomposable into two or more related threads. Jiao discloses shader programs for 3D computer graphics and graphics shader operations that are translated into a plurality of threads. Ex. 1005, 1:47-55, 2:37-53. Wilt discloses commands related to and/or defining processing tasks and processing tasks relating to a wide variety of applications including filtering of video and image rendering operations (e.g., vertex shader, geometry shader, and/or pixel shader programs). Ex. 1004, 3:60-64, 4:7-10, 33-40. Wilt discloses that the term “thread” refers to an instance of a particular (shader) program executing on a particular set of input data and that SIMD or SIMT instruction issue techniques may be used to support parallel execution of a large number of threads. *Id.* 5:61-6:7. Therefore, Wilt discloses that shader programs are decomposed into two or more related threads .

4. It is true that because a task can be comprised of one or more threads, a task and a thread are not technically the same. However, a system that saves, sends, or executes all of the threads that make up a task, means that system has saved, sent, or executed the corresponding task. In addition, a task may also be represented in other ways, such as a pointer or index to the constituent one or more threads that

make up the task. In other words, tasks need not necessarily comprises threads. For that reason, it was my understanding that Judge Elliot rejected the Patent Owner's proposed construction that would define tasks in terms of its constituent one or more threads. Ex. 2003, 37 ("Nor need the task necessarily comprise thread or shader programs, because the specification states that those are optional...").

5. Further, in my opinion, Dr. Hart's inclusion of the term "related" introduces ambiguity by restricting the construction to "related" threads. A POSITA could reach different conclusions regarding what was meant by the term "related." For example, must the two or more threads be "related" to each other or just to the task?

6. Finally, Dr. Hart's proposed construction is inconsistent with the '381 patent and would exclude its preferred embodiments as understood by a POSITA. Based on Dr. Hart's proposed construction, the claimed tasks can only be the construct that exists before it is decomposed, *i.e.*, "a unit of processing work decomposable into two or more related threads." This would mean that the plurality of tasks received by the shader core in claim 1 must be the exact same pre-decomposed task received by the engines. That is not how a POSITA would understand the '381 patent disclosure. The claimed engines will process the tasks it receives and what is then sent to the shader core will be altered and no longer in its pre-decomposed construct. During his deposition, Dr. Hart realized his mistake and

tried to say that threads over time can be a task as long as there is some “correspondence” to the task. However, he was not able to show what “correspondence” he meant, where “correspondence” was required in the claims. Ex. 1009, 174:3-181:15.

7. In my opinion, Dr. Hart’s proposed construction should be rejected.

B. “engine”

8. In my opinion, the claims do not “explicitly recite” that “each” engine must be configured “to independently” receive tasks and load state data. Just the opposite. A POSITA would understand that the plain language of Claim 15 is that the “plurality of engines” are treated as a unit or group. The “plurality of engines” receives a plurality of tasks and loads state data. Nothing in the claim requires that “each” engine must “independently” be configured to perform all the recited functions in the claim. Indeed, the shader core need only be “configured to receive the plurality of tasks from at least one of the plurality of engines.” (emphasis added). In other words, a single engine can receive the plurality of tasks and send the entire plurality to the shader core for the claim to be satisfied. This would mean that it was a single engine that “received” the plurality of task. Further, it would allow a different engine within the plurality to load the state data.

C. “configured to”

9. My understanding of “configured to” is that it means more than just capable of. For example, I was informed that “configure to” could imply a system that is designed or set up for operation in a particular way to perform the recited function on more than mere chance. And I prepared my opinions based on that understanding. However, in my opinion, the prior art shows such designs and set-ups, such that the prior art would have been understood to be “configured” as claimed. *See, e.g.*, Ex. 2007, 109:17-25; 157:4-15, Ex. 1003, 79, 80, 99, 111, 125, and so on. To the extent any such “configuration” is not explicit, it is obvious based on the fact that a POSITA would have understood the benefits of having the prior art systems “configured” in that manner for all the reasons in the Petition and my declarations, and my deposition testimony.

D. “each of the plurality of tasks”

10. This is the same argument that Patent Owner asserts for “engine,” that “each individual” engine must perform each recited function. As explained above, the proposed limitations are not consistent with the plain language of the claims.

11. Contrary to Dr. Hart’s suggestion, I have never taken the position that that each engine is required to load state data for each of the pluralities of task that are received. Moreover, it is my understanding that Dr. Hart’s proposed construction was rejected by the ITC court. Ex. 2003, 97-98, 153, 174.

III. ALL OF THE CHALLENGED CLAIMS ARE OBVIOUS

A. All Challenged Claims

1. Wilt

a. **Wilt Renders Obvious a Shader Core that Simultaneously Executes Different Tasks [Limitation 15[e]]**

12. Wilt discloses simultaneous execution of at least two tasks in the shader core, under two alternative theories. Pet., 10-12, 24-25, 28. In the first alternative, which the Patent Owner identifies as the “single PPU theory,” the “engines” are the front-end unit 212, work distribution unit 210, core interface 303 within a single PPU 202 whereas the “shader core” is the set of cores 208 within the single PPU. The cores are organized in the SIMT fashion, which means that the concurrent threads running on a single PPU of Wilt can still “follow divergent execution paths,” i.e., these concurrent threads (which are predicated with an implicit task) will be producing results for different sets of instructions on different control-flow paths. Therefore, under the “single PPU theory,” Wilt discloses a “shader core” that effectively performs two or more different instruction streams corresponding to two or more different tasks simultaneously.

13. In the second alternative, which Dr. Hart identifies as the “multiple PPU theory,” the front-end unit 212, work distribution unit 210, and core interface 303 within a single PPU 202 constitute a single engine, and the collection of these single engines constitute the claimed “engines.” Therefore, the set of all cores 208 within

all PPUs 202 of the parallel processing system 112 correspond to the claimed “shader core.” The Petition explains how the cores 208 receive processing tasks to be executed via a work distribution unit 210 and can be programmed (independently) to execute different types of processing tasks relating to a wide variety of applications, including data transforms, filtering of video and/or audio data, modeling operations, image rendering operations (e.g., vertex shader, geometry shader, and/or pixel shader programs), and so on. Pet., 10-12, 24-25, 28; Ex. 1004, 4:1-23, 33-45. Further, Wilt discloses that some embodiments of the parallel processing subsystem comprise PPUs for graphics processing and PPUs for general-purpose processing, again, two different types of tasks. *Id.*, 4:46-60, 8:56-67; Ex. 1003, ¶48.

14. Thus, under both theories, a POSITA would have understood that each PPU is designed to execute two different tasks simultaneously, *i.e.*, two different programmable processing cores 208 within a PPU can take divergent execution paths, and therefore, perform different tasks¹; or, even if the processing cores within a single PPU are focused on a single task, the core sets within different PPUs are designed to execute different tasks. *Id.* At a minimum, a POSITA would have

¹ Consider a very simple example of an instruction stream running on a SIMT core set: “thread_id = threadIdx.x; IF (thread_id == even) THEN execute task A ELSE execute task B. This instruction stream can be mapped to cores 208 within a single PPU, resulting in half of the cores executing task A while the other half executing task B.

understood such specific designs to be obvious in light of the benefits associated with running multiple concurrent tasks on a graphics processing system containing a large number of processing cores. These benefits, which would be well known to a POSITA, include higher core utilization and higher system performance (lower latency and higher processing throughput).

15. In my opinion, all of Patent Owner and Dr. Hart's arguments or opinions related to "task" are predicated on his incorrect claim construction that a task must be "decomposable" (i.e., it must be in the pre-decomposed construct) and that it must be "decomposable into two or more related threads." Ex. 2006, ¶70 ("Wilt's cores 208 do not actually receive and execute tasks as claimed by the '381 Patent because the highest order item that Wilt's system receives are commands or command streams (*i.e.*, threads)") ¶76 ("Dr. Pedram has not shown that the system is 'configured to' execute threads that were decomposed from two separate parent tasks"), 44 ("Nor can it be assumed that any two (or more) threads will correspond..."). Because that presumptive construction is wrong, these arguments are largely irrelevant.

16. Patent Owner and Dr. Hart alleges that even if Wilt discloses simultaneous execution of different threads, which represent tasks, nothing indicates those could be from different tasks. POR, 40-41, 43; Ex. 2006 ¶¶69-76. Regardless, Dr. Hart argues that Wilt's shader core is not configured to execute a first task while

executing a second task. POR, 41; Ex. 2006 ¶74. As explained above, each core is designed to be programmable. Wilt expressly discloses a design with programmable cores that can take divergent execution paths within a single PPU and can execute two or more different types of tasks on different PPUs, e.g., general purpose or graphics processing. Pet., 10-12, 24-25, 28; Ex. 1003, 4:1-23, 33-45. At a minimum, this disclosure renders the claim limitation obvious.

17. Dr. Hart alleges that Wilt's cores do not receive "tasks" at all, but instead command or command streams, which are, at best, threads or subcomponents of tasks. POR 41-43; Ex. 2006 ¶¶70-71. However, Wilt expressly discloses that cores 208 receive tasks by receiving commands defining the processing tasks. Ex. 1003, 4:7-10 "Cores 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." *See also Id.*, 4:15-17 "When a new processing task arrives, work distribution unit 210 assigns the task to a core 208 that is asserting the ready signal ...". These express disclosures are fatal to Patent Owner on this issue. Nonetheless, Dr. Pedram has testified that a single thread could be a task. Ex. 2007, 11:7-16. Further, Wilt discloses "the term 'thread' refers to an instance of a particular program executing on a particular set of input data." Ex. 1003, 5:63-65. A POSITA would understand that Wilt's "thread" is an instance of a particular program (i.e., a sequence of tasks-defining commands) executing on a particular set of input data.

Thus, Wilt's threads meet the requirements of a task as proposed by Patent Owner in the ITC case and adopted by the ALJ, i.e., "a task is a unit of processing work or portion of an application to be executed." *See* Section II.A (discussion of "task").

18. Patent Owner incorrectly states that I "could not, under oath, state that a thread can be a task." This is objectively false because I expressly 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.** I mean, there's no requirement that a task results in multiple threads. **Task could be a single thread.** ... depending on the context, a thread itself might be a simple one or a very complicated one.") (emphasis added). The number of threads that would be decomposed from a task is simply a matter of degree, e.g., simple to complex tasks will range from one thread to many. See my discussion of task in the claim construction section above.

19. Finally, Patent Owner, recognizing that Wilt clearly discloses that different types of tasks are carried out in its processing cores, either within a PPU or in different PPUs, makes a last-ditch effort by focusing on the term "configure to." Wilt's disclosure is for a system that is designed or set up for operation in a particular way to perform the recited function on more than mere chance. For example, the cores are programmable which a POSITA would understand to mean a computational component within a device or system that can be customized and

programmed to perform specific functions or tasks. This core typically executes a set of instructions or code that can be modified to meet the requirements of different applications. Also, Wilt expressly discloses that the cores carry out numerous different types of tasks. Pet., 10-12, 24-25, 28; Ex. 1001, 4:1-23, 33-45. A POSITA would understand those tasks are carried out simultaneously because of the knowledge of a POSITA about the SIMT organization of the cores within a single PPU and explicit disclosures in Wilt about the independent execution of multiple tasks on different PPUs. At a minimum, a POSITA would have understood the benefits of carrying them out simultaneously, such as increased core utilization, lower end-to-end latency (measured as the time difference between when a task arrives until the task gets completely executed) and higher processing throughput (defined as the number of tasks that are executed in a unit of time e.g., a second) and, thus, found such a configuration obvious when practicing Wilt.

20. In summary, a POSITA would understand that Wilt is specifically designed, configured, to be a highly parallel system. Each PPU is in parallel to every other PPU. Each PPU is identical in design. Each PPU can execute different types of tasks, such as pixel or vertex or general compute or graphics processing type tasks. Further, a POSITA would understand that it would be desired to have and use such a configuration to increase core utilization, lower end-to-end latency, and increase processing throughput.

b. A Shader Core Comprised of the Plurality of Wilt's PPU's Discloses and Renders the Challenged Claims Obvious

21. Patent Owner and Dr. Hart argue that the front end, work distribution unit, and core interface of a single PPU cannot collectively constitute a claimed “engine.” POR 45; Ex. 2006 ¶77. Dr. Hart states that he would not consider these three items as “a single item” and that no POSITA would group them together. *Id.* However, these three components meet the requirements of a claimed engine. Pet., 26; Ex. 1003, ¶71. Tellingly, Patent Owner or Dr. Hart never argue or offer any explanation or analysis that the “engine” I have identified, comprised of the front end, work distribution unit, and core interface of a single PPU, do not perform the same function as the claimed engines or that they do not meet the construction of engine (under either of the proposed construction).

22. Second, Patent Owner and Dr. Hart argue that the cores across multiple PPU's cannot be the claimed “shader core” because (a) they are not related to each other, (b) they cross PPU boundaries, (c) they have no “point-to-point” connections, (d) I could not opine that PPU's communicate with each other,² (e) they receive commands from the engines within its own PPU, and (f) it will not have the required “resource sharing” and “engine relationship.” POR, 46-47; Ex. 2006 ¶78. In my

² This again is a mischaracterization of Dr. Pedram's testimony. His actual testimony is that although PPU's in the parallel processing subsystem 112 do not appear to directly communicate with each other, they can communicate with each other through their memory interfaces 214 and common memory bridge 105. Ex. 2007, 42:21-45:6.

opinion, none of these “distinctions” are claim elements or restrictions imposed on the shader core by the claims.

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

23. Patent Owner’s hyper-technical argument that Wilt’s engines cannot be engines because they are inside the PPU drawn box in Wilt (and thus inside the shader core) is form over substance. Although I generally referred to the PPU as the shader core, I also provided a detailed explanation of what components actually comprised the functionality for each claim element and, in doing so, acknowledged the “separateness” of the claim elements Patent Owner and Dr. Hart conspicuously overlook. Pet, 28; Ex. 1003 ¶¶71, 73, 75-77. Specifically, for both the single- and multi-PPU embodiments, I identified the engines as separate components from the shader core that perform the function required by the claims, i.e., receiving tasks and loading state data. *Id.*, ¶¶71-73. Later, when addressing the shader core element, I was clear that the processing cores 208 “execute concurrent tasks” received from the engines. *Id.*, ¶¶75-77.

2. Jiao

a. Jiao Discloses and Renders Obvious a “Plurality of Engines” as Claimed

(1) Jiao’s Shaders Receive Tasks

24. I previously explained that a POSITA would have understood Jiao’s shaders receive high level graphics tasks and produce low level shader tasks that are

mapped to threads running on Jiao's shader core. Pet. 40-43, 47-48; Ex. 1005, 9:43-49, 5:47-51. This is consistent with Jiao's disclosure that it "communicates instruction sets or commands through the use of the CPU 126 to registers in the GPU 114." I explained that a POSITA would have understood that Jiao's commands are tasks as claimed. Ex. 1003, ¶104.

25. Further, the commands must make their way to Jiao's engines and subsequently to Jiao's shader cores. Pet., 44-46. Patent Owner ignores the obvious: the fact that each of the vertex, geometry and pixel engines of Jiao can perform many different functions shows that these shaders receive commands ("tasks") that tells them which functions they must perform. Also, as the Board recognized, Jiao's engines "include internal registers and various caches for configuring their fixed function graphics hardware units for operations in computational core 204." ID, 27. A POSITA reviewing the entirety of Jiao would have understood Jiao's engines to receive the claimed tasks even if that is not expressly stated.

(2) Jiao's Shaders Are Not Part of the Shader Core

26. I never intended the engines (Jiao's shaders) to be part of the shader core. It seems a hyper-technical argument on the part of Dr. Hart. I addressed the engines separately from the shader core limitation and as such it was inherent and obvious that they were not intended as part of the shader core, which is a separate and distinct limitation of the claims.

b. Jiao's Scheduling Module is Not Associated with the First Processing Unit.

27. In my opinion, Patent Owner and Dr. Hart's interpretation of this claim limitation directly contradicts the '381 patent and would fail to cover its own embodiments disclosed therein. As shown in Figure 4 of the '381 patent, there are only two processing units identified, CPU 102 and GPU 110. Ex. 1001, 10:5-6. GPU 110 is the "first processing unit," e.g., it comprises the plurality of virtual engines 432A-432N. Thus, CPU 102 is the "second processing unit." The OS scheduler 310 is the "scheduling module" that sends tasks to the virtual engines. *Id.*, 10:25-32. As shown, the OS scheduler 310 is not "within" the second processing unit (CPU 102). Further, a POSITA reading the '381 patent would have understood that the OS scheduler 310 resides in the system memory 104 of the computer system 100 and is executed on CPU 102 as system software.

c. Jiao's Shader Core Receives the Plurality of Tasks

28. Dr. Hart is wrong that the Jiao's shader core receives the plurality of tasks before the engines. As I previously explained, Jiao's engines receive high level tasks and produce low level tasks, which a POSITA would understand to be the threads corresponding to the high-level tasks, that are then mapped to threads of the execution engines. Ex. 1003, ¶ 101.

29. The fact that some of the tasks do not undergo the logic transformation in the '381 specification does not subtract from the fact that many tasks do go

through this transformation, which shows that the tasks received by the shader core were not required to be the same tasks (construct) received by the engines, *i.e.*, that Dr. Hart's proposed construction of task is incorrect.

30. Patent Owner and Dr. Hart's argument that the '381 input logic does not generate tasks, as Jiao's shaders do, nor do they process them, is just wrong. The input logic includes "graphics pre-processing logic (which prepares graphics-processing tasks for issuance to shader core 214)." '381 at 7:30-33. All of these components, e.g., graphics pre-processing logic, vertex analyzer, and scan converter, process or modify the tasks received by Jiao's shaders (*i.e.*, engines) before they are sent to Jiao's shader core. Specifically, the graphics pre-processing logic identifies "shader programs associated with a graphics-processing and/or general-compute task," generates "pointers to a vertex buffer" to prevent redundant processing of vertices, provides "connectivity data" for proper rasterization of primitives, and traverses primitives to determine "pixels to be processed by shader core." *Id.* at 7:34-62. These actions illustrate the processing of input tasks and the generation of corresponding output tasks, which include distinguishing between graphics-processing and general-compute tasks, directing each task to the suitable resource, preventing duplicate processing of vertices, providing necessary connectivity data for primitive rasterization, and identifying pixels needing shader core processing.

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

31. I have already responded to Patent Owner and Dr. Hart's repeated attempt to equate Jiao's tasks to "what the '381 Patent would call threads." *See, e.g.*, Section III.A.2.c. above. Patent Owner's only twist here is the allegation that because, in Jiao, tasks can be associated with a single context, they cannot be the claimed tasks. POR, 62. However, it is not true that concurrently running tasks in Jiao's system are only associated with a single context. Jiao expressly states: "execution of shader programs with constants of a **different context occurs in simultaneous paths**, with one path corresponding to the previous context constants and at least one other path corresponding to the new context constants. Ex. 1005, 3:53-60, 8:65-9:9:14, 10:14-54. The updating or replacement of constants through implementation of the various CB system embodiments enables a smooth procedure of constant replacement in shader execution blocks, which may result in an improvement in computational efficiency." Ex. 1005, 3:53:60. Jiao's disclosure that its tasks can be linked to multiple contexts is fatal to Patent Owner and Dr. Hart's argument that Jiao's tasks are equivalent to the threads described in the '381 Patent.

3. Stuttard

a. Stuttard Discloses Sending and Simultaneously Executing Tasks

32. Stuttard's plurality of engines (channel controller 108, array controller 104, and thread manager components cache memory unit 1024, thread processors

1026, and status block 1030) and shader core (plurality of processing blocks 106) receive tasks. Pet., 55-60; Ex. 1003 ¶¶ 123-128.

(1) Stuttard's Engines and Shader Core Receive and Process Tasks

33. Dr. Hart is wrong that Stuttard's engines and shader core to no receive and/or process tasks.

34. A task is a unit of processing work or a portion of an application to be executed. In Stuttard, the thread manager 102 uses control signals from EPU 8 to retrieve instructions that are provided to array controller 104 and channel controller 108, which in turn transfer the control signals to processing blocks 106 and array 1061 of the processing elements to execute. Stuttard explains that these instructions are fetched from the local memory 12 or the EPU 8 by the fetch unit 1023 and supplied to the cache memory 1024 in the thread manager 102 via connecting logic. Ex. 1006, 11:12-14. Thread processors 1026 control the issuance of core instructions from the thread manager 102 to the array controller 104 and channel controller 108 to keep processing blocks 106 active. Ex. 1006, 11:27-12:7 and 26:3-7. Thus, instructions provided by EPU 8 are received by the asserted engines 104, 108, 1024, 1026 and make their way to the shader core. Stuttard explains that "instruction streams for the processing blocks 106 are known as 'threads'" and that "each thread works cooperatively with other threads to perform a task or tasks." Stuttard explains that in this way his system can support both "multi-threading" and "multitasking."

Ex. 1006, 10:7-17. As explained above, instructions that comprise a thread are fetched by the fetch unit 1023 and written to the cache memory 1024, and subsequently, dispatched to one or more processing blocks 106. Therefore, what is received by Stuttard's "engines" and what is received by Stuttard's "shader core" are simply instructions that comprise one or more threads, where these threads can correspond to a (single) task or (multiple) tasks as expressly disclosed in Stuttard. As both the Board and ITC recognized, these instruction streams meet the definition of tasks, in that, they are a unit of processing work.

35. Patent Owner and Dr. Hart provide no analysis of any kind why the identified tasks sent from EPU 8 to the engines and shader core, do not meet the construction of task proposed by Patent Owner and adopted at the ITC. To the extent such argument is based on its proposed construction requiring tasks to be decomposable into two or more threads (POR 65, "Thus, both thread scheduler and EPU, at best, are capable of only activating one thread for execution." (emphasis added)), such construction is improper. *See* Section II.A. To the extent Patent Owner's argument is based on having intermediate components, i.e., the engines and/or shader core does not directly receive tasks, it is wrong. There is nothing in the claims that requires any such construction. Ex. 1001, 5:4-13, 10:14-39, Fig. 4. The '381 expressly disclosed non-direct routing of tasks. *Id.* At a minimum, a

POSITA would have understood these disclosures in Stuttard to render obvious that its engines and shader core is receiving tasks.

(2) Stuttard's Simultaneously Executes Different Threads

36. Contrary to Dr. Hart's allegation, I did not rely on just one sentence in Stuttard to show simultaneous execution of different tasks. The thrust of my argument is that Stuttard must be taken as a whole. This is a reoccurring problem with Patent Owner and Dr. Hart, that they are focusing on a particular example instead of Stuttard as a whole.

37. Stuttard explicitly discloses parallel threads assigned to different tasks at the same time on processing blocks 106 as a shader core. Ex. 1006, 11:19-25. Stuttard discloses threads may perform a (single) task or (multiple) tasks or execute multiple tasks according to their priority levels. *Id.* 10:7-12, 23:35-25:17 Stuttard discloses high vs low priority threads. I explained that high priority tasks create high priority threads and vice versa. Ex. 2007, 152:7-14. Stuttard's explicit teaching that it can run 8 parallel threads that are assigned to different tasks at the same time directly refutes Patent Owner and Dr. Hart's statement of "no disclosure" by Stuttard. It also refutes any argument that Stuttard can process only a single task at one time. Ex. 1006, 11:19-25. In short, it is my opinion, and it should be beyond dispute, that Stuttard system is configured to execute two or more tasks simultaneously.

38. Patent Owner and Dr. Hart's suggestion that the purpose for Stuttard's partitioning is so that it can operate on different regions of the display or is just for redundancy is unpersuasive. A POSITA would understand that operating on different regions would have been achieved at lower hardware cost by having a single processing block, 8 times the size of each of the 8 smaller blocks, being fed by a single thread processor 1026. There are no benefits to partitioning a large processing block into 8 smaller processing blocks supported by 8 different thread processors 1026 if all these smaller blocks do is to run the exact same thread on different data. While there is a teaching about redundancy and fault tolerant design in Stuttard, the idea is to add one additional redundant PE for each panel of 32 PEs. Ex. 1006, 9:22-10:3. With this (fine-grain) redundancy scheme, there will be no benefit in having one big processing block versus having 8 parallel smaller blocks as long as the big array or the smaller blocks are each decomposed into these panels with one additional redundant PEs per panel. A POSITA would know that the fine-grain redundant design such as what is taught in Stuttard is better (in terms of area and performance of the resulting system) than a coarse-grain redundancy scheme where one or more processing blocks are used as redundant blocks to provide redundancy and fault tolerance for Stuttard's system.

39. In summary, a POSITA would understand the claimed configuration to be obvious in light of Stuttard even if not expressly disclosed due to, at least, (1) the

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

b. Stuttard's Engines are Not Within the Shader Core

40. While I did include Stuttard's binning unit as one possible engine in the plurality of engines I identified in Stuttard, I never relied on the binning unit in any of my analysis. Even without the binning unit as one of the plurality of engines, the remaining engines still constitute a plurality and, as I explained in detail in my initial declaration, those remaining engines carry out the claimed functions for the plurality of engines.

B. Claim 19

1. Wilt

a. Petitioner's "Single PPU" Theory Invalidates Claim 19

41. Patent Owner and Dr. Hart's sole objection to what they call the "single PPU" theory where the cores within one PPU is the claimed shader core, is that the cores within a single PPU cannot do execute both graphics-processing and general compute tasks at the same time. Untrue. I explained that "each parallel processing unit (PPU) ... execute[s] concurrent tasks [*plural*] (using its processing cores 208)."

Ex. 1003 ¶ 76. Further, I explained that Wilt teaches a plurality of PPUs and walked through the components within the representative PPU, that is, they are “identical” in design and have the same functionality. *See, e.g., Id.* ¶ 47. Wilt expressly states that each PPU is identical, and Dr. Hart admitted this in his deposition. Ex. 1004, 4:46-60; Ex. 1009, 158:8-15. I explained that Wilt discloses that some PPUs are designated to carry out graphics processing tasks, while other PPUs are designated to carry out general compute tasks. Ex. 1003, ¶ 88. Because they are identical, the fact that some are assigned graphics processing task while others are assigned general compute tasks, means they are each configured to execute both types of tasks. Further, Wilt expressly discloses that each core 208 carries out SIMT execution, which “allows different threads” to “follow divergent execution paths.” Ex. 1004, 5:61-6:12; Pet., 12. Thus, a POSITA would have understood that the concurrent threads running on a single PPU of Wilt can “follow divergent execution paths” which means that at some points in time, they will be executing different instructions, *i.e.*, two different tasks at the same time. Further, because Wilt discloses that each PPU can do either graphics-processing and general-compute tasks, a POSITA would have understood each PPU was configured to execute those two types of tasks as needed in the “divergent execution paths.”

b. Petitioner’s “Multiple PPU” Theory Invalidates Claim 19

42. As I explained, each of Wilt's PPUs can execute "concurrent tasks," either within a single PPU or in two or more PPUs. The Petition shows that Wilt discloses the concurrent tasks can be graphic-processing and general-compute tasks. To the extent not expressly disclosed, a POSITA would have understood the limitations of claim 19 to be part of Wilt's disclosed design (as described above and in the Petition), and/or that it was obvious based on those disclosures. At a minimum, it was obvious to configure Wilt's system to simultaneously execute the two tasks that Wilt expressly discloses are executed to achieve the advantages discussed herein and in the Petition. *See, e.g.*, Pet. 10.

2. Jiao

43. Dr. Hart's sole argument is that Jiao does not expressly disclose general-compute tasks. However, Petitioner asserts an obvious ground. I explained that Jiao's shader core executes multiple simultaneous tasks. Pet., 47-48; Ex. 1003 ¶¶ 110-111. Jiao discloses "arithmetic logic units and other logic," which would have been understood to execute general-compute tasks. Pet., 51; Ex. 1003 ¶ 117. Further, the described functionality of "instruction decoding, operand fetch, branch computation, instruction execution, shift logic, load store, among other processing functionality or logic" would be understood to be general-compute tasks. *Id.*; Ex. 1005, 6:1-48, 11:4-13; Ex. 2007, 113:17-116:18. Further, a POSITA would have understood and found it obvious that Jiao's programmable shaders (304, 306, and

310) “can be used to do both graphics-processing and general-compute tasks.” Ex. 1003 ¶ 117. Thus, it is my opinion that a POSITA would have considered Jiao to disclose, and/or render obvious in light of the disclosure, execution of general-compute tasks. Pet., 51; Ex. 1003 ¶ 117. Stated differently, a POSITA would have found it obvious to use Jiao’s programmable core to perform graphics-processing and general-compute tasks simultaneously.

44. Patent Owner’s allegation that I confirmed that the tasks generated by Jiao’s shaders would not include general compute tasks is false and intentionally misleading. POR, 33. I was expressly asked if the functions in Jiao at 11:4-13, a reference expressly about the EUDP, were being performed by Jiao’s shaders (claimed “engines”). The obvious answer is no. My response had nothing to do with what Jiao’s EU Pool (EUP) 414 (claimed “shader core”) actually executes.

45. Patent Owner and Dr. Hart’s next argument is that the ALU functionality is not within the shader core. However, Jiao discloses that the EU data path (EUDP 514) is part of the EU pool (EUP) 414, *i.e.*, the shader core. Ex. 1005, 10:33-39 and Ex. 2007, 119:6-19. At a minimum, a POSITA would have understood that it would have been obvious. Patent Owner’s reference to Dr. Pedram’s testimony regarding whether or not the EUPD 514 can comprise functional elements 420 and Dr. Hart’s allegations about modifying EUP 414 is irrelevant. The question

is whether or not EUDP is part of the shader core (EUP 414), which I confirmed. Ex. 2007, 119:6-19.

3. Stuttard

46. I testified that a “general-compute task” is “something other than a graphics processing task,” which could include things like “standard fetch and decode, standard execution, write-back to memory, load/store, ALU operations, addition, multiplication, shift, so on...” (Ex. 2007, 113:17-25); that “a computational task does involve IO operations.” (*Id.* 150:7-13); and that Stuttard’s general system control is a type of general compute task, and/or a POSITA would have found it obvious to be such (Ex. 1003 ¶ 70). Further, a POSITA would recognize (if it is not inherent) that an instruction stream, executed on a processor block, will perform computations.

47. In casting Stuttard’s I/O operations as non-computational tasks, Patent Owner omits my direct testimony that “[I/O actions] could happen during – as part of a computational task.” Ex. 2007, 151:5-6. Further, Stuttard’s discussion relates to general I/O operations that may be part of a 3D activity, a 2D activity, or a general system control task. Ex. 1006, 10:24-31. Consistent with this disclosure in Stuttard, I also testified that “a computational task does involve I/O operations.” Ex. 2007, 150:12-15. A POSITA would know that I/O operations support the execution of both graphics processing and general compute tasks. Ex. 2007, 148:15-150:19. In

that sense I/O operations by themselves are neither graphics processing tasks nor general compute tasks but are evidence in Stuttard that there are overlapping operations with high efficiency that is only possible if multiple threads corresponding to I/O operations and general system control, 2D activities, and 3D activities can run at the same time. Even if Stuttard is found not to explicitly state “general control system tasks are general compute tasks,” a POSITA would have found it obvious because they would have understood the benefit and necessity of using such types of tasks along with the graphics processing tasks (e.g., 3D graphics tasks) to achieve the “scale and richness of computational resources” in Stuttard’s computing system. Pet. 61.

C. Claim 20

1. Wilt

48. I do not agree that Wilt’s context swapping does not “necessarily involve or require the use of task priorities.” As I previously explained, a POSITA would have known that Wilt’s disclosure of an efficient context swapping scheme for PPU threads within the same CPU thread can support prioritization of PPU threads such that the execution of a lower priority PPU thread can be preempted in favor of executing a higher priority PPU thread. Ex. 1004, Abstract and 12:4-13:15; Ex. 2007, 60:6-9, 61:20-62:1.

49. Further, Patent Owner and Dr. Hart's distinction regarding threads and tasks is without merit. Wilt's CPU thread is invoking the parallel processing unit (PPU) to perform various graphics processing tasks as part of the CPU thread. Ex. 10011:25-29 ("Each CPU thread 450 within the user application 410 may perform independent tasks for the user application 410, where each task requires a different PPU context."). Thus, swapping PPU contexts means swapping tasks that make use of their associated contexts.

50. Thus, as I explained, a POSITA reading Wilt's teachings would understand that Wilt's efficient context swapping mechanism is needed for the purpose of bringing higher priority tasks to the forefront so that they can promptly run on the available compute resources. Ex. 2007, 59:6-62:16.

2. Jiao

51. Patent Owner and Dr. Hart are wrong that Jiao's priorities are not related to tasks and/or would not impact tasks. While Jiao may not expressly state the obvious, a POSITA would understand that Jiao's disclosure of a priority level associated with the rendering context would support efficient service of tasks with a priority level. This is because a task with a higher priority would require an associated higher priority rendering context if it is to run faster (i.e., with higher priority) than the low priority tasks. Conversely, a high priority rendering context associated with a higher priority task would enable higher priority execution of the

said task. A POSITA would have realized that it is wasteful (non-beneficial) to have a high priority rendering context that uses Jiao's fast access path to be associated with a low priority task. This is because the fast access path is more expensive to implement (from a hardware cost perspective) to be realized than a slow access path. Similarly, a POSITA would have found it counterproductive (disadvantageous) to have a high priority task with a low priority rendering context.

52. Further, the rendering contexts are provided before the tasks belonging to the context are commenced. Ex. 1005, 9:5-7. Thus, a task is not sent to the execution unit until and unless the corresponding rendering context has been loaded. *Id.*, 13:32-53, 13:63-14:46. And it is clear, and not disputed by Dr. Hart, that the constants are loaded based on a priority. *Id.* Thus, the tasks associated with the rendering contexts are loaded based on a priority associated with the tasks.

3. Stuttard

53. Stuttard expressly discloses that its threads are assigned priorities related to one another. Ex. 1003 136; Ex. 100611:16-25, 23:3-26:10. Even based on Dr. Hart's incorrect construction of task, he cannot deny that it is obvious that priority of the constituent threads of a task will impact how the task will be received based on those priorities.

54. Patent Owner concedes that Stuttard discloses threads with different priority levels (e.g., high or low) and using priority of threads to determine their

order of execution. The only issue is that Patent Owner claims that threads are not tasks. However, at a minimum, a POSITA would understand a task would comprise one or more threads and a high-priority task would give rise to high-priority threads. Ex. 1003 ¶ 137. Conversely, once all high-priority threads associated with a task are executed based on their assigned priority levels, the task has received a high priority execution. I testified, “the fact remains that if a task is high priority, a high-priority task, the threads that it spawns to be executed, those threads also would be high-priority tasks. Whether they are all exactly the same priority or different priorities, I don't think Stuttard has discussed that. It would be more of an implementation detail.” Ex. 2007, 152:7-14. In other words, the threads resulting from the decomposition of a high priority task will all be high priority. They may not have the exact same high priority level assignment as the parent task but that is irrelevant.

55. Finally, Patent Owner and Dr. Hart selects a sentence from Stuttard that says a “yielding” thread may yield to another thread if paused for a time. However, a POSITA would understand Stuttard’s system can be configured so that a high priority thread may not assume a yielding status. For example, in the same paragraph quoted by Patent Owner and Dr. Hart, Stuttard also indicates that high priority threads should not be treated as yielding. Ex. 1006, 24:30-32. Stuttard also expressly states that during a yielding event, the scheduler activates the highest priority thread that is ready, i.e., “selection based on a priority associated with the

task.” *Id.* 25:5-10. Thus, even if true that a high-priority thread is paused and replaced by another thread, as soon as the cause for the high priority thread’s pause is gone (i.e., the thread is again ready to run), the high-priority thread will preempt any existing thread and run again. That is all the claim requires.

IV. DECLARATION

56. In signing this declaration, I understand that this declaration will be filed as evidence in a contested case before the patent Trial and Appeal Board of the USPTO.

57. I am aware that willful false statements and the like made by me in connection with this declaration are punishable by fine or imprisonment, or both (18 U.S.C. § 1001) and may jeopardize the validity of this Reply and/or the Petition. All statements herein made by me of my own knowledge are true and all statements made on information and belief are believed to be true. I declare under penalty of perjury that the foregoing is true and correct.

Executed this 10th day of April, 2024 at Los Angeles, California.

Respectfully Submitted,



Massoud Pedram, Ph.D.