

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

REALTEK SEMICONDUCTOR CORP.,  
Petitioner,

v.

ATI TECHNOLOGIES ULC  
Patent Owner.

---

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

---

**DECLARATION OF JOHN HART REGARDING  
U.S. PATENT NO. 8,854,381**

TABLE OF CONTENTS

|                       | <u>Page</u> |
|-----------------------|-------------|
| I. INTRODUCTION ..... | 1           |
| II. Wilt .....        | 1           |
| A. All Claims.....    | 1           |
| B. Claim 19 .....     | 2           |
| III. Jiao .....       | 3           |
| A. All Claims.....    | 3           |
| B. Claim 19 .....     | 6           |
| C. Claim 20 .....     | 7           |
| IV. Stuttard.....     | 8           |
| A. All Claims.....    | 8           |
| B. Claim 19 .....     | 8           |
| V. CONCLUSION.....    | 9           |

I, John Hart, being of legal age, hereby declare, affirm, and state the following:

## **I. INTRODUCTION**

1. This declaration is made in support of Patent Owner's Sur-Reply. I respond to the technical issues raised in Petitioner's Reply and Dr. Pedram's Reply Declaration, Ex. 1012. Nothing in Petitioner's Reply or Dr. Pedram's Reply Declaration has changed any of my opinions that the challenged claims of the '381 patent are valid.

2. I understand Petitioner claimed that I had no opinions under the ITC's construction of "task" and that my opinions applied only to my construction of "task." That is untrue, my opinions apply regardless of the construction of "task." That is, my opinions provided in my prior declaration (Ex. 2006) made in support of Patent Owner's Response apply equally even if task is construed as being decomposed into one or more threads.

## **II. Wilt**

### **A. All Claims**

3. The "SI" of SIMT stands for "single instruction." The "MT" means that thread-based state data can be used to mask the instruction so that its effects are nullified on some of the PEs. This means that the instructions are not necessarily performed on all of the data in a data group, but only some. This provides flexibility in processing over SIMD where every instruction is acted on all the data. Thus,

SIMT does not mean two different instructions are being executed simultaneously—by definition it is the same instruction.

4. Wilt’s statement that it can be programmed to execute a “wide variety of applications” is not indicative of whether Wilt can execute two tasks at the same time. Ex. 1004, 4:33-40. All this says is Wilt has the ability to execute tasks relating to different applications, not that Wilt is executing any of these tasks simultaneously.

**B. Claim 19**

5. Wilt’s disclosure of two embodiments, one “optimized for graphics and video processing” and the other “optimized for general purpose processing” does not disclose a system which can execute both graphics-processing tasks and general-compute tasks as required by claim 19. Even if Wilt’s system was capable of doing so, there is nothing in Wilt that indicates it can execute graphics-processing tasks and general-compute tasks at the same time, also as required by claim 19. Wilt’s disclosure of general purpose processing, of which there is only one sentence, is insufficient to lead a POSITA to believe Wilt is configured to meet the requirements of claim 19, or is even capable of doing so. Wilt would need to provide much more details to describe a system that could process both types of required tasks at the same time. A POSITA would not believe the authors of Wilt believed they had invented that type of system given the minimal, non-enabling statements.

6. Even if Dr. Pedram's opinions were correct that Wilt could process both types of tasks in one embodiment (which I disagree with), this only discloses mere capability, not actual configuration, which is required. In order to show actual configuration, for example, one would need to show disclosures in which engines are receiving both types of tasks, and sending to the shader core both types of tasks for processing simultaneously. But Dr. Pedram did not, and thus fails to prove the required configuration is disclosed.

7. Wilt's statement that PPUs "may be operated in parallel to process data at a higher throughput than is possible with a single PPU" does not state anything regarding whether it is capable of processing graphics-processing tasks and general-compute tasks at the same time. It only provides that operating multiple PPUs in parallel provides greater throughput.

### **III. Jiao**

#### **A. All Claims**

8. The only disclosure Jiao has regarding where the commands go is that they are communicated to "registers in the GPU 114." Ex. 1005, 5:17-23. Jiao provides nothing further, including any indication that these specific registers are within the shaders. Jiao at 5:47-51 does not support tasks being received by the shaders. It merely states the computational core 204 is capable of meeting the computational requirements imposed by the shader tasks. Jiao at 9:41-49 also does

not support tasks being received by the shaders. It 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. In contrast, Jiao provides ample detail in how the shaders all receive data, and what the shaders do with that data. If the shaders received tasks, Jiao would say so, just like it explains in great detail how the shaders receive data.

9. The '381 patent does not support finding what is generated by Jiao's driver must be received by Jiao's shaders. All 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.

10. Dr. Pedram opines that 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. That is incorrect. The '381 patent does not state the OS scheduler resides in system memory. *E.g.*, Ex. 1001, 6:42-50, 10:22-32. Even if it did reside in system memory as code, which appears to be Petitioner's argument, the OS scheduler is the scheduler as run, not merely sitting in memory. In order 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." The fact that Fig. 4 of the '381 patent depicts the

ATI Ex. 2009

IPR2023-00687

Page 6 of 12

OS schedule as “outside” of the CPU does not 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.

11. Jiao does not support finding concurrently running tasks (“shader programs”) may be associated with multiple rendering contexts in each execution unit. Dr. Pedram identifies this disclosure of Jiao: “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. It is wrong to interpret this to mean that different tasks are running concurrently. The context around this sentence in Jiao describes updates and changes to constants and relates to the constant buffer system (or CB system). For example, the sentence before the one quoted above by Dr. Pedram states: “In particular, the below described CB systems enable updates or changes to constants to occur while enabling execution of previous constants.” Thus, there is no disclosure that Jiao’s EUs are “simultaneously running multiple contexts.” Ex. 1005, 10:38-39. Instead, the other part cited by Dr. Pedram 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, and has nothing to do with whether or not the EUs can simultaneously run multiple tasks.

**B. Claim 19**

12. Dr. Pedram’s opinion that “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 is incorrect. These are actions that pertain to all instructions/threads, and indicative of whether a system is capable of executing general-compute tasks. For example, instruction decode helps translate the instruction. Operand fetch gets operand values from the memory or registers. Instruction execute conducts an operation with the operands to get some result. Load store would be to load something from memory into a register, a store is to store something from a register into a memory address. Thus, nothing about this functionality is so specific to general-compute tasks, and exclusive of graphics-processing tasks, such that a POSITA would understand Jiao could execute general-compute tasks from this disclosure.

13. Jiao’s disclosure of “programmable” shaders are not indicative of general-compute tasks. Jiao does not say that the vertex, geometry, and pixel shaders are mere examples. Rather, Jiao says: “The computational core 204 comprises a pool of multiple execution units to meet the computing requirements imposed by shader tasks associated with various shader programs, including a vertex shader,

geometry shader, and/or pixel shader, processing data for the graphics pipeline 208.” Ex. 1005, 5:47-61. Jiao has another similar statement: “Various types of shader programs can be executed or mapped to the computational core 204, including the vertex shader 304, geometry shader 306, and pixel shader 310.” *Id.*, 6:59-62. Jiao does not identify any other types of shaders. Therefore, a POSITA would understand that the vertex, geometry, and pixel shaders are what Jiao means when referring to “shader programs.” Jiao never says that the vertex, geometry, and pixel shaders are mere examples.

14. Dr. Pedram claims Jiao discloses EUDP is part of the alleged shader core, EUP 414, citing Jiao at 10:33-39. This disclosure discusses the location of the fast access path, and not the EUDP, and only provides that “*fast access path 506* is implemented in computational core EUs 420 (e.g., via an EU data path (EUDP, 514), described below) and provides fast access to constants (1 cycle delay).” Ex. 1005, 10:33-39. A POSITA would understand that “*via* EUDP,” means 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, but is not part of the shader core.

### **C. Claim 20**

15. Dr. Pedram opines that rendering contexts are provided before the associated task commences execution. This is in no way indicative that the rendering contexts are priorities associated with the tasks, as required. Even if tasks are not

sent to the execution unit unless the corresponding rendering context has been loaded, this only makes loading the rendering context a pre-requisite to executing the task. In fact, it is a necessary step prior to executing any task. It does not disclose task priority.

#### **IV. Stuttard**

##### **A. All Claims**

16. To the extent Dr. Pedram asserts one thread can be decomposed from two different tasks, this is incorrect. Stuttard's statement "[e]ach thread works co-operatively with other threads to perform a task or tasks" means the threads *altogether* are working together to perform a task or multiple tasks, not a single thread can perform multiple tasks. Ex. 1006, 10:10-12.

##### **B. Claim 19**

17. There are other types of tasks besides graphics-processing tasks and general-compute tasks. Stuttard, for example, provides there are general system control tasks. So it is incorrect to say that any task that is not a graphics-processing task is a general-compute task. Even Dr. Pedram acknowledged there are some tasks that don't involve computations. 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). I/O tasks are not general-

compute tasks. While I/O actions may be part of a general-compute task, that does not make them general-compute tasks by themselves.

## **V. CONCLUSION**

18. In signing this declaration, I recognize that it will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place in the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

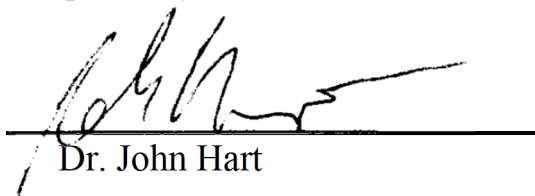
19. The statements and opinions in this declaration comprise and are based upon my review, analysis and understanding of the source materials that I have reviewed as discussed herein, based on the disclosure and statements made within such source materials.

20. I reserve the right to supplement my opinions in the future to respond to any arguments that the Petitioner raises and to take into account new information as it becomes available to me.

21. I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

May 22, 2024

Respectfully Submitted,



Dr. John Hart