

Exhibit 6

**UNITED STATES INTERNATIONAL TRADE COMMISSION
WASHINGTON D.C.**

**Before the Honorable Cameron Elliot
Administrative Law Judge**

In the Matter of

**CERTAIN GRAPHICS SYSTEMS,
COMPONENTS THEREOF, AND
DIGITAL TELEVISIONS CONTAINING
THE SAME**

Investigation No. 337-TA-1318

**DECLARATION OF DR. JOHN C. HART IN SUPPORT OF COMPLAINANTS
ADVANCED MICRO DEVICES, INC. AND ATI TECHNOLOGIES ULC'S
INITIAL MARKMAN BRIEF**

August 15, 2022



Dr. John C. Hart

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. EXPERT QUALIFICATIONS	1
III. MATERIALS CONSIDERED	3
IV. PERSON OF ORDINARY SKILL IN THE ART.....	3
V. BACKGROUND TECHNOLOGY	4
VI. The ‘053 Patent	4
A. Summary	4
B. Asserted Claims of the ‘053 Patent.....	6
C. “arbiter”	7
D. “command processing engine”	8
E. “command threads”	9
F. “memory device”	10
G. “coupled to”	11
VII. The ‘381 Patent	12
A. Summary	12
B. The Asserted Claims of the ‘381 Patent	18
C. “shader core”	19
D. “task”	23
E. “engine” / “plurality of engines”	24
F. “load”	26

TABLE OF EXHIBITS USED

Exhibit No	Document
1	U.S. Patent No. 7,742,053
4	U.S. Patent No 8,854,381
7	U.S. Pat. App. No. 12/874,134, Reply to Final Office Action of Jan. 14, 2014, dated Mar. 13, 2014
8	U.S. Patent App. No. 16/257,500 Amendment to the Claims, dated July 12, 2021
9	U.S. Pat. App. No. 12/874,134, Amendment, dated Oct. 18, 2013
10	Arbiter Definition, <i>Electrical Engineering Dictionary</i> (2000)
11	Arbiter Definition, <i>IEEE 100 The Authoritative Dictionary of IEEE Standard Terms, Seventh Edition</i> (2000)
13	Command Definition, <i>Microsoft Computer Dictionary</i> (2002)
14	Couple Definition, <i>IEEE 100 The Authoritative Dictionary of IEEE Standard Terms, Seventh Edition</i> (2000)
23	Memory, <i>Electrical Engineering Dictionary</i> (2000)
34	Arithmetic Logic Unit Definition, <i>Microsoft Computer Dictionary</i> (2002)
35	Pat Hanrahan & Jim Lawson, <i>A Language for Shading and Lighting Calculations</i> , 24 SIGGRAPH '90 at 289 (1990)
36	Arithmetic and Logic Unit Definition, <i>Dictionary of Computer Science, Engineering, and Technology</i> (2001)
37	Thread Definition, <i>Microsoft Computer Dictionary</i> (2002)
38	David Luebke & Greg Humphreys, <i>How GPUs Work</i> , 40 IEEE COMPUTER 96 (2007)
39	Peter N. Glaskowsky, <i>NVIDIA's Fermi: The First Complete GPU Computing Architecture</i> (2009)
40	Task Definition, <i>The American Heritage Dictionary</i> (4th ed)
41	Task Definition, <i>IEEE 100 The Authoritative Dictionary of IEEE Standard Terms, Seventh Edition</i> (2000)
42	Task Definition, <i>Webster's New World Computer Dictionary</i> (10th ed.)
43	Task Definition, <i>Oxford Dictionary of Current English</i> (1996)
44	Task Definition, <i>Concise Oxford American Dictionary</i> (2006)
45	Task Definition, <i>Microsoft Computer Dictionary</i> (2002)
46	Task Definition, <i>American Heritage College Dictionary</i> (2007)
47	Excerpt from Jim X. Chen & Chunyang Chen, <i>Foundations of 3D Graphics Programming</i> (2d ed. 2008) (https://www.google.com/books/edition/Foundations_of_3D_Graphics_Programming/UJlpzmCpPXcC?hl=en&gbpv=1)

I. INTRODUCTION

1. I, Dr. John C. Hart, have been retained by counsel for Complainant Advanced Micro Devices, Inc. (“Complainant” or “AMD”) as an expert to provide this declaration and certain opinions regarding certain claim terms relating to United States Patent Nos. 7,742,053 (“the ‘053 Patent”), and 8,854,381 (“the ‘381 Patent”). In particular, I have been asked to analyze how a person of ordinary skill in the art would understand certain claim terms. I have personal knowledge of all the facts set forth herein, and if called to testify at any hearing or trial in these cases, I would competently testify and verify that contained herein.

2. I am being compensated at my hourly rate of \$575. I am also being separately reimbursed for out of pocket expenses. My compensation does not depend in any way on the outcome of these cases or the particular testimony or opinions I express.

3. I understand that Complainant is asserting claims 1-4, 5-9 of the ‘053 Patent and claims 15, 16-20 of the ‘381 Patent against each of the Respondents in this Investigation.

II. EXPERT QUALIFICATIONS

4. I have been a professor of computer graphics since 1992. I am currently a tenured Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign, a department consistently ranked in the top-5 by US News and World Report. I am also Director of the Office for Professional Education of the University of Illinois at Urbana-Champaign.

5. I received a Bachelor’s of Science degree in Computer Science from Aurora University in 1987, a Master’s of Science in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1989, and a Ph.D. in Electrical Engineering and Computer Science from the University of Illinois at Chicago in 1991.

6. I worked as an intern in 1998 at the IBM TJ Watson Research Center in Hawthorn,

New York, and in 1999 at AT&T Pixel Machines in Holmdel, New Jersey. I worked as a postdoc in 1991-1992 at the Electronic Visualization Laboratory at the University of Illinois at Chicago with funding from the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. I worked as a visiting researcher in the summer of 2007 for the Graphics Research Group of Adobe Systems in Seattle, Washington.

7. From 1997-2000, I designed, simulated and patented graphics hardware for Silicon Reality and the Evans & Sutherland Computer Corp. I have also consulted for visual effects companies (e.g., Kleiser-Walczak, Blue Sky | VIFX), defense contractors (e.g., SAIC, Pratt & Whitney) and medical imaging companies (e.g., Intrinsic Medical Imaging).

8. My research in computer graphics has been supported by Adobe, DARPA, Intel, Nokia, NVIDIA, NSF and Microsoft. I have published over 75 reviewed papers on computer graphics, including papers in the most rigorous and prestigious venues, including the SIGGRAPH Annual Conference Proceedings, the ACM Transactions on Graphics, and the IEEE Transactions on Visualization and Computer Graphics. From 2002-2008, I was the longest-serving Editor-in-Chief of ACM Transactions on Graphics, the top journal in the field of computer graphics.

9. I was an Executive Producer for the ACM SIGGRAPH documentary “The Story of Computer Graphics” in 1999. I served from 1996-2000 on the ACM SIGGRAPH Executive Committee. I am currently the computer graphics area editor of the ACM Books series, and oversaw the recent publication of “The VR Book: Human Centered Design for Virtual Reality.”

10. I co-authored the book “Real-Time Shading” in 2002 on the implementation of procedural shaders on graphics processors because they became fully user programmable GPU’s. I contributed two chapters to the book “Modeling and Texturing: A Procedural Approach” in 2002, including a chapter on procedural texturing using rasterization and texturing operations.

11. My CV is attached as **Exhibit A**.

III. MATERIALS CONSIDERED

12. In forming the opinions set forth in this declaration, I have considered and relied upon my education, knowledge of the relevant field, and experience. I have also reviewed and considered the Asserted Patents, their prosecution histories, the prior art, Complainants' and Respondents' proposed lists of claim terms for construction and constructions, the Joint Claim Construction Chart submitted by the Parties, and other materials expressly cited.

13. I understand that I may be called on to testify in this matter at a claim construction hearing. I therefore reserve the right to consider and/or rely upon any additional information or materials that may be provided to me or that are relied upon by any of the Respondents' experts or witnesses.

14. I understand that discovery in this Investigation is still ongoing, and I may consider additional facts and materials produced through discovery to determine whether such additional materials have an impact on my opinions.

IV. PERSON OF ORDINARY SKILL IN THE ART

15. The field of the '053 and '381 Patents is computer graphics hardware. Professionals in the field of computer graphics hardware will generally be electrical engineers, computer engineers, or computer scientists.

16. It is my opinion that a person of ordinary skill in the art of computer graphics hardware would have a degree in electrical engineering, computer engineering, computer science, or a related field. Such a person would also have at least 2-4 years' experience working in computer graphics hardware/computer architecture or related fields, or an equivalent combination of graduate education and/or work experience.

17. I meet this criteria and I consider myself a person with at least ordinary skill in the

art pertaining to the '053 and '381 Patents. In addition, I would have been a person of ordinary skill at the time of the inventions of the '053 and '381 Patents.

V. BACKGROUND TECHNOLOGY

18. I have reviewed the Background Technology section from the Declaration of Dr. Anselmo Lastra in Support of AMD's Initial Markman Brief. I agree with the opinions expressed in that section of Dr. Lastra's Declaration. That Background Technology applies equally to the '053 and '381 Patents and I incorporate that Background Technology section herein in its entirety.

VI. THE '053 PATENT

A. Summary

19. The '053 Patent is directed to graphics processing systems, and specifically towards the processing multiple command threads in parallel, which is also known as multi-threading. Additionally, the '053 Patent uses a unified command processing engine which can process both vertex command threads and pixel command threads.

20. In graphics processing systems, the management and control of multiple command threads relating to texture applications is important. '053 Patent at 1:22-24. Typically, multiple steps are required to processes elements (e.g., vertex and pixel command threads) for the application of textures and other processing structures. *Id.* at 1:24-28. A preferred manner for improving the efficiency of graphics processing systems is controlling the flow of the multiple command threads. *Id.* at 1:28-31.

21. According to the '053 Patent, prior art systems (as shown in Fig. 1) were rigid systems that required providing command threads sequentially. For example, such systems included multiple arbiters and "specifically delineated [arithmetic logic units] ALU resource buffers and texture fetch resource buffers." *E.g., Id.* at 1:32-35, 1:38-40, 1:65-67, Fig. 1. Typically, "the buffers are first in and first out (FIFO) buffers," and the "command threads must

be sequentially provided between the various buffers.” *Id.* at 1:35-36, 1:65-2:2. The prior art embodiments further provide that command threads in the ALU resource division are provided to an ALU, and the command threads within the texture fetch resource division are provided to a texture fetch processor, in contrast to unified shader processor described by the invention of the ‘053 Patent. *Id.* at 1:42-48; *see Id.* at Abstract. Additionally, prior art systems did not “support an unlimited number of dependent fetches based on the structure of the buffer[s]” and “connectivity between each other and with respect to available ALU resources and texture fetch resources.” *Id.* at 2:2-6. (A “dependent texture fetch” occurs when two or more texture fetches are made, and the result of an earlier texture fetch is used to modify the location of a subsequent texture fetch.)

22. The invention of the ‘053 Patent allows for a “multi-thread graphics processing system” including reservation station that is capable of storing both pixel command threads and vertex command threads, and a “command processing engine” that is “capable of processing [both] pixel command threads and vertex command threads.” *Id.* at 2:36-45, Abstract. The system described by the ‘053 Patent introduces a far more flexible approach that ensures greater efficiency, ease, and speed.

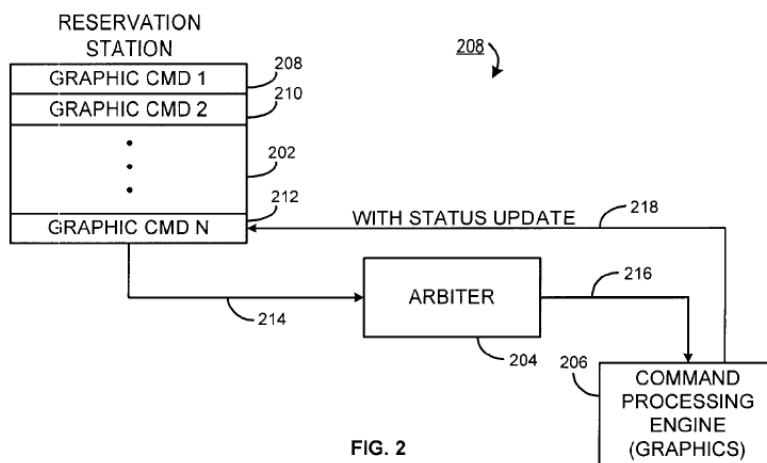


FIG. 2

23. For example, the embodiment in Fig. 2 illustrates a graphics processing system with “a reservation station 202, an arbiter 204, and a command processing engine 206.” *Id.* at 2:63-67.

In this embodiment, “[t]he arbiter 204 retrieves a command thread via connection 214” from the plurality of command threads in reservation station 202 “and provides the retrieved command thread to the command processing engine 206, such as a graphics processing engine via connection 216.” *Id.* at 2:67-3:11. “Thereupon, the command processing engine 206 performs a threaded command and provides a status update 218 to the reservation station 202, more specifically to the corresponding command thread, such as 208, 210 or 212.” *Id.* at 3:23-15.

B. Asserted Claims of the ‘053 Patent

24. AMD asserts claims 1 through 9 of the ‘053 Patent, reproduced below, with **bold** over the claims terms at-issue in this *Markman* brief as shown below.

1. A graphics processing system comprising:

at least one **memory device** comprising a first portion operative to store a plurality of pixel **command threads** and a second portion operative to store a plurality of vertex **command threads**; and

an **arbiter, coupled to** the at least one **memory device**, operable to **select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads** based on **relative priorities** of the plurality of pixel **command threads** and the plurality of vertex **command threads**.
2. The graphics processing system of claim 1, further comprising:

a **command processing engine, coupled to** the **arbiter**, wherein the **arbiter** is further operable to provide the command thread to the **command processing engine**.
3. The graphics processing system of claim 1, wherein the first portion of the at least one **memory device** comprises a first **memory device** and the second portion of the at least one **memory device** comprises a second **memory device**.
4. The graphics processing system of claim 1, wherein a single **memory device** comprises the first portion and the second portion of the at least one **memory device**.
5. A graphics processing system comprising:

at least one **memory device** comprising a first portion operative to store a plurality of pixel **command threads** and a second portion operative to store a plurality of vertex **command threads**;

an **arbiter**, coupled to the at least one **memory device**, operable to **select a command thread from either of the plurality of pixel command threads and the plurality of vertex command threads**; and

a plurality of **command processing engines**, coupled to the **arbiter**, each operable to receive and process the command thread.

6. The graphics processing system of claim 5, wherein the plurality of **command processing engines** comprises at least one arithmetic logic unit.

7. The graphics processing system of claim 5, wherein the plurality of **command processing engines** comprises at least one texture processing engine.

8. The graphics processing system of claim 5, wherein the first portion of the at least one **memory device** comprises a first **memory device** and the second portion of the at least one **memory device** comprises a second **memory device**.

9. The graphics processing system of claim 5, wherein a single **memory device** comprises the first portion and the second portion of the at least one **memory device**.

C. “arbiter”

25. A POSITA would have understood the term “arbiter” by its plain and ordinary meaning.

26. I understand that Respondents have proposed that “arbiter” be construed to mean a “component that actively selects one of a plurality of command threads based on state bits associated with each command thread.” I disagree with their construction.

27. Arbiter is a term that is commonly used in the art and known to a POSITA. Various definitions of the term “arbiter” accurately describe the meaning including the following:

- “a unit that decides when multiple requestors may have access to a shared resource. Ex. 10, Electrical Engineering Dictionary (2000);
- “[t]he module that is performing the arbitration,” i.e., “the process of determining which requesting device will gain access to a resource.” Ex. 11, IEEE (2000).

28. Respondents propose by their construction to restrict the arbiter to select “one of a

plurality of command threads based on **state bits** associated with each command thread.” (emphasis added). A POSITA would understand that the ‘053 Patent discloses some embodiments of the arbiter that utilize state bits (*see e.g.* ‘053 Patent at 4:49-5:18) and other embodiments of the arbiter that do not require state bits (*see e.g.* ‘053 Patent at 4:28-48). The arbiter does not have to make a selection based on the state bits.

29. There are embodiments of the arbiter that utilize both state bits and status bits. *See* ‘053 Patent at 4:49-5:18 (“The bits required for the thread arbitration may be stored in a highly multi-ported structure, such that the bit stored in the one read port device are terms state bits and the bits stored in the multi-read port device are termed status bits.”). “In one embodiment the state bit includes, but is not limited to, a control flow instruction pointer, a loop iterater, a call return pointer, predicated bits, a GPR base pointer, a context pointer, valid bits, and any other suitable bits as recognized by one having skill in the art.” ‘053 at 4:57-61 (emphasis added). A POSITA would have understood from the disclosure of these particular embodiments that these state bits were intended to be utilized by the processor of the thread, and not the arbiter of the thread.

D. “command processing engine”

30. A POSITA would have understood the phrase “command processing engine” by its plain and ordinary meaning.

31. I understand that Respondents have proposed that “command processing engine” be construed to mean a “software or hardware module that processes commands.” I disagree with their construction.

32. A POSITA would have understood the term “engine” as a configuration of hardware to perform a particular task. The specification describes “[t]he command processing engine may be any suitable engine as recognized by one having ordinary skill in the art for processing commands, such as a texture engine, an arithmetic logic unit, or any other suitable

processing engine.” ‘053 Patent at 2:59-62. A POSITA would understand that an arithmetic logic unit (ALU) is implemented in hardware, not software. Ex. 34, Microsoft Computer Dictionary (2002) (“**arithmetic logic unit** *n.* A component of a microprocessor chip used for arithmetic, comparative, and logical functions. Acronym: ALU.”); Ex. 35, Electrical Engineering Dictionary (2000) (“**arithmetic-logic unit** *See* arithmetic and logic unit.”); Ex. 35, Electrical Engineering Dictionary (2000) (“**arithmetic and logic unit** a combinational logic circuit that can perform basic arithmetic and logical operations on n-bit binary operands.”); Ex. 36, Dictionary of Computer Science, Engineering, and Technology (2001) (“**arithmetic-logic unit** *See* arithmetic and logic unit.”); Ex. 36, Dictionary of Computer Science, Engineering, and Technology (2001) (“**arithmetic and logic unit** a combinational logic circuit that can perform basic arithmetic and logical operations on n-bit binary operands.”). A person of ordinary skill in the art would further understand that a “texture engine” in the context of a GPU denotes hardware. Further, as is commonly used in the art, the term engine, which appears in texture engine and processing engine, connotes hardware structure.

33. A POSITA would understand that the ‘053 Patent discloses a system general realized in hardware. In isolated circumstances, the specification specifically describes that certain elements can be implemented in software. One example is the disclosure that “[t]he arbiter may be any implementation of hardware, software or combination.” ‘053 Patent, 2:48-52. I believe that is the only element of the disclosure that is described as being potentially implemented in software.

E. “command threads”

34. A POSITA would have understood the phrase “command threads” by its plain and ordinary meaning. To the extent that the phrase “command threads” requires additional explanation, it should be interpreted as “streams of instructions or processes that are parts of a larger process or program.”

35. I understand that Respondents have proposed that “command threads” be construed to mean “a sequence of commands, instructions, and state bits required to process a vertex or pixel element” I disagree with their construction.

F. “memory device”

36. A POSITA would have understood the phrase “memory device” by its plain and ordinary meaning.

37. I understand that Respondents have proposed that “memory device” be construed to mean “memory with dedicated storage areas for pixel and vertex command threads.” I disagree with their construction.

38. A POSITA would understand the term “memory” to have its plain meaning. The following definitions are consistent with a POSITA’s understanding of the term “memory”:

- “area for storing computer instructions and data for either short-term or long-term purposes.” Ex. 23, Electrical Engineering Dictionary (2000)

39. A POSITA would have understood the term “memory” to be any type of memory. As provided by the ‘053 Patent, “[a] reservation station may be *any type of memory device* capable of reserving and storing command threads.” ‘053 Patent at 2:39-41 (emphasis added).

40. A POSITA would not consider a “memory device” to require “dedicated storage areas for pixel and vertex command threads.” The term “dedicated” does not appear anywhere in the ‘053 patent. A POSITA would have understood that the “memory device” could store pixel and vertex command threads anywhere within the memory.

41. A POSITA would understand the term “command” to have its plain meaning. The following definitions are consistent with a POSITA’s understanding of the term “command”:

- “instruction to a computer program that, when issued by the user, causes an

action to be carried out.” Ex. 13, Microsoft Computer Dictionary (2002).

42. A POSITA would understand the term “thread” to have its plain meaning. The following definitions are consistent with a POSITA’s understanding of the term “thread”:

- “a process that is part of a larger process or program.” Ex. 37, Microsoft Computer Dictionary (2002).

43. A POSITA would have understood the phrase “command threads” as used in the ‘053 Patent maintains its plain meaning. Specifically, the ‘053 Patent specification states that “a command thread is a sequence of commands applicable to the corresponding element, such as [a] pixel command thread relative to processing of pixel elements and a vertex command thread relative to vertex processing commands.” ‘053 Patent at 2:41-45.

G. “coupled to”

44. A POSITA would have understood the phrase “coupled to” by its plain and ordinary meaning.

45. I understand that Respondents have proposed that “coupled to” be construed to mean “directly connected” I disagree with their construction.

46. A POSITA would understand the term “coupled to” to have its plain meaning. The following definitions are consistent with a POSITA’s understanding of the term “coupled to”:

- “The association of two or more circuits or systems in such a way that power or signal information may be transferred from one to another.” Ex. 14, IEEE (2000)

47. A POSITA would understand that the different elements of the disclosure need to be connected to communicate information, but direct connection is not necessary. As described by the ‘053 Patent, “[t]he arbiter 204 retrieves a command thread via connection 214 and provides

the retrieved command thread to the command processing engine 206, such as a graphics processing engine via connection 216.” ‘053 Patent at 3:8-11. The ‘053 Patent describes information flow through “coupled to” connections. The term direct, directly or the like does not appear in the specification.

VII. THE ‘381 PATENT

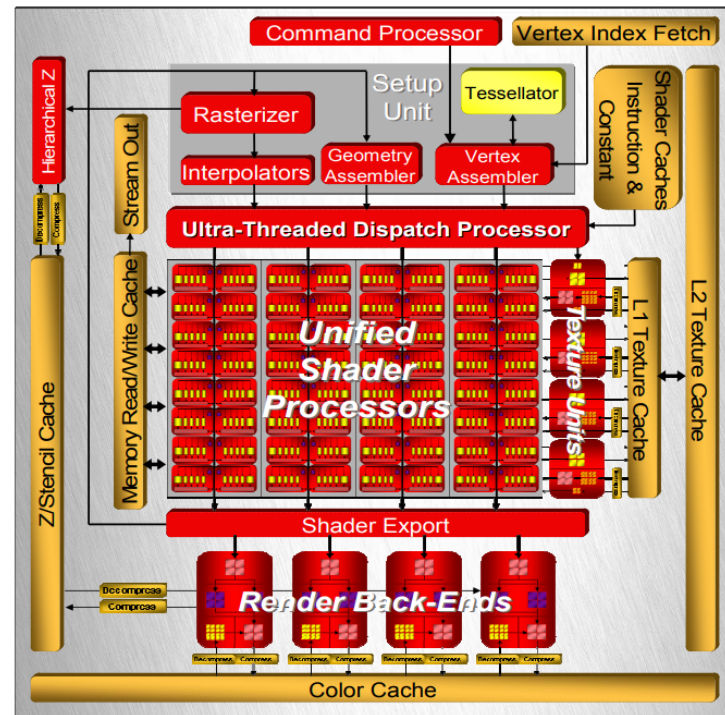
A. Summary

48. The ‘381 patent is directed to graphics processing units, and specifically toward the specialized GPU hardware for the scheduling of work to facilitate the parallel execution of graphics processes.

49. An important aspect of this parallelism is its utilization, ensuring these parallel processors are continually provided with code to execute, to reduce the time they would otherwise spend idle. However, submitting all of that work to the GPU and scheduling it efficiently on the available processing resources is still a complex problem. GPUs rely heavily on pipelining to achieve their performance, and bottlenecks at any stage of the pipeline can leave other stages under-utilized. A graphics application such as a videogame can also involve the processing of multiple concurrent tasks each with their own multistage pipelines. Each stage of each pipeline of each task can make extensive use of parallel processing, but the throughput of each stage can also depend on the priority and readiness of other stages.

50. Parallel computers strive to optimize the utilization and throughput of their processors through careful scheduling, which allocates computational resources to the execution of tasks. Before the ‘381 patent, GPUs contained specialized circuitry for very fine grain scheduling of processing of tasks submitted by a graphics driver associated with a host CPU. The architecture of AMD’s Radeon X2900XT GPU shown below, for example, incorporated an “ultra threaded dispatch processor” circuit, which divided up tasks as they came in from the host CPU

driver into threads, and scheduled those threads for execution on the unified shader processors. (These shader processors are called “unified” because these processors are designed to operate on both “vertices” and “pixels.”)



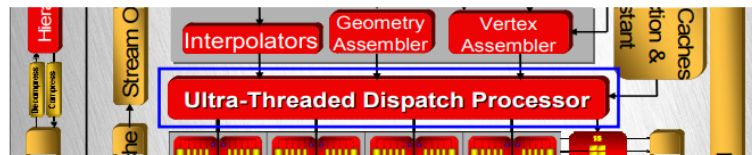
https://fileadmin.cs.lth.se/cs/Personal/Michael_Doggett/talks/2007_UMR_R2900.pdf.

51. The ultra-threaded dispatch processor served as the main control for the shader core and allowed for hundreds of threads to be “in flight” or simultaneously executed by the same shader core. If there were 100 processors available to process these one million threads, then this ultra-threaded dispatch processor would be responsible for assigning each processor its portion of 10,000 threads to run. Having many threads (e.g. 10,000) available to a single processor is helpful because if that processor has to wait for one thread to e.g. access memory, it can run instructions from another thread while it is waiting.

Ultra-Threaded Dispatch Processor



- Main control for the shader core
 - All workloads have threads of 64 elements
 - 100's of threads in flight
 - Threads are put to sleep when they request a slow responding resource
- Arbitration policy
 - Age/Need/Availability
 - When in doubt favor pixels
 - Programmable



25

September 2007 Radeon HD 2900 and Geometry Generation

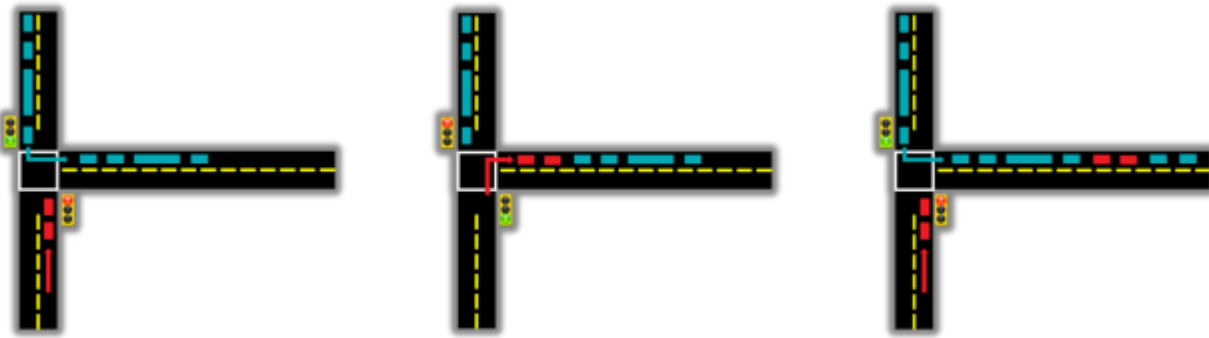
Id.

52. As the inventors of the '381 patent recognized, however, “[t]o pass tasks to the GPU, an operating-system (OS) scheduler stores the tasks in a command buffer. A conventional GPU processes one command buffer at a time. The OS scheduler serially places tasks in the command buffer, and the GPU typically processes the tasks in the order in which they are placed in the command buffer.” “In some instances, however, the GPU may process tasks out of the order in which they were placed in the command buffer. For example, the GPU may interrupt the execution of a first task to execute a more-important (e.g., low-latency) task that was placed in the command buffer after the first task.”

53. “Although context switching allows a GPU to process tasks out of the order in which they were placed in the command buffer, context switching is problematic for several reasons. As an initial matter, a substantial amount of time is required to perform a context switch, thereby limiting the performance of the GPU. Also, context switching requires additional local

memory (e.g., back-up storage units) to store the context that is being switched. The additional local memory takes up precious chip area, resulting in a larger GPU.”

54. The example below further illustrates the problem. Consider the example below, where two streams of traffic (representing task queues) are attempting to merge onto a freeway (representing GPU processing resources). A simple way of handling this is with traffic signals, which allow one traffic stream to enter the freeway while the other waits in a queue. Periodically the light switches, allowing some traffic from both streams onto the freeway.

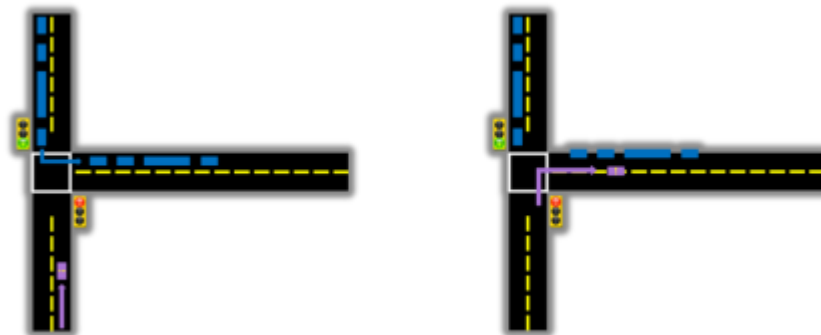


55. To get the GPU to switch from working on one task to another, a number of steps are required:

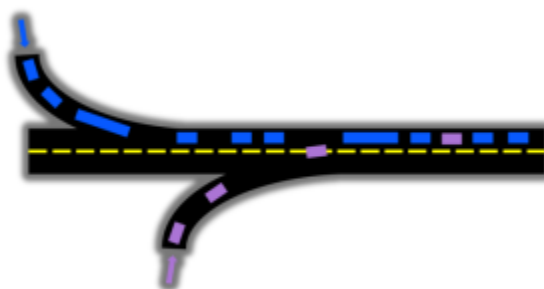
1. Stop submitting new work associated with the current task
2. Allow all calculations in flight to complete
3. Replace all context data from the current task with that for the new task
4. Begin submitting work associated with the new task

56. An alternative way to handle scheduling is by assigning priorities to each queue, and allowing tasks in higher priority queues to pre-empt those in lower priority queues. Pre-emption means that a lower priority task can be temporarily suspended while a higher priority task completes. Continuing with the traffic analogy, high priority tasks are treated like emergency vehicles – that is, they have right-of-way at intersections even when the traffic light is red, and

other vehicles on the road must pull to the side to let them pass.



57. Yet a third approach is to allow new tasks to begin executing without having to suspend tasks already in flight. This requires the ability to perform fine-grained scheduling and interleaving of tasks from multiple queues. The mechanism would operate like on-ramps merging on to a freeway, where there are no traffic signals and vehicles merge directly without forcing anyone to stop and wait.



58. The ‘381 patent discloses and claims the GPU architecture needed to enable this third approach. As illustrated below, in the annotated asserted claim and Fig. 2 of the ‘381 patent, claim 15 is drawn toward a computer system apparatus comprising a memory (“Memory System” highlighted in green), first processing unit (“GPU” highlighted in pink), bus (highlighted in light blue), plurality of engines (highlighted in teal), and shader core (highlighted in red). The ‘381 patent specification provides further elaboration on how the claimed engines enable substantially simultaneous execution of tasks on a shader core. “Command processor 230 receives tasks (e.g., graphics-processing and general-compute tasks) from one or more command buffers filled by the

OS scheduler[,]" and incorporates a number of engines "that share resources of GPU 110[.]" As illustrated in Fig. 2, the shared GPU resource is the "shader core" 214. 75.

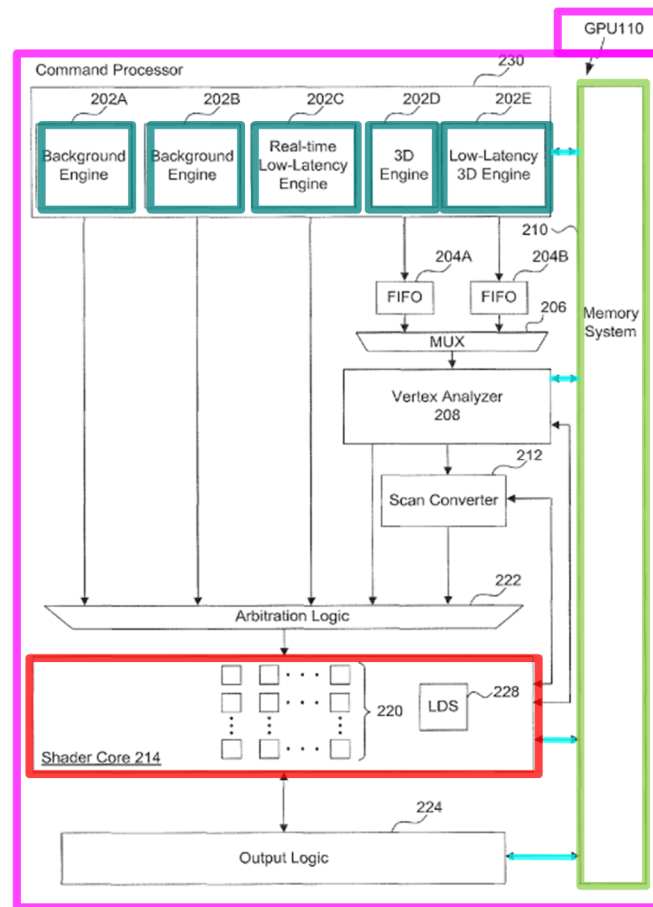


FIG. 2

59. A computing system comprising:
- a memory;
 - a first processing unit; and
 - a bus coupled to the memory and the first processing unit, wherein the first processing unit comprises:
 - a plurality of engines configured to receive, from a scheduling module associated with a second processing unit, a plurality of tasks and to load state data associated with each of the plurality of tasks; and
 - a shader core configured to receive the plurality of tasks from at least one of the plurality of engines and to execute a first task from the plurality of tasks while executing a second task from the plurality of tasks based on respective state data

associated with each of the first and second tasks.

60. This defining characteristic of the individual engines – their ability to share a single shader core and allow tasks to be independently executed on single shader core – is one of the key features of the shader that enables the GPU to perform fine grain scheduling and interleaving of tasks from multiple queues.

B. The Asserted Claims of the ‘381 Patent

61. AMD asserts claims 15 through 20 of the ‘381 Patent, reproduced below, with **bold** over the claims terms at-issue in this *Markman* brief:

62. A computing system comprising:

a memory;

a first processing unit; and

a bus coupled to the memory and the first processing unit, wherein the first processing unit comprises:

a **plurality of engines** configured to receive, from a scheduling module associated with a second processing unit, a plurality of **tasks** and to **load** state data associated with each of the plurality of **tasks**; and

a **shader core** configured to receive the plurality of **tasks** from at least one of the **plurality of engines** and to execute a first task from the plurality of **tasks** while executing a second task from the plurality of **tasks** based on respective state data associated with each of the first and second **tasks**.

63. The computing system of claim 15, wherein the **shader core** comprises:

a first plurality of processing elements configured to execute the first task based on a first set of state data associated with the first task; and

a second plurality of processing elements configured to execute the second task based on a second set of state data associated with the second task.

64. The computing system of claim 15, wherein the **plurality of engines** comprises a first queue configured to receive the first task and a second queue configured to receive the second task.

65. The computing system of claim 15, wherein the first task comprises a low-latency task and the second task comprises a regular-latency task.

66. The computing system of claim 15, wherein the first task comprises a graphics-processing task and the second task comprises a general-compute task.

67. The computing system of claim 15, wherein the **shader core** is configured to receive the plurality of **tasks** based on a priority associated with each of the plurality of **tasks**.

C. “shader core”

68. A POSITA would have understood the phrase “shader core” by its plain and ordinary meaning. If a construction is given, I agree with AMD’s construction, which is “graphics processing portion of the first processing unit circuitry comprising a plurality of processing elements for execution of various threads (e.g., shader programs).”

69. I understand that Respondents have proposed that “shader core” be construed to mean “an array of processing elements and associated components for executing threads.” I disagree with their construction.

70. A “core” as used in the industry, refers to a processing unit, containing circuitry to execute instructions. The term “shader” connotes instructions designed for graphics processing.

71. A POSITA would understand the term “shader core” to have its plain meaning. The following definition is consistent with a POSITA’s understanding of the term “shader core”:

- “With the unified architecture, a single floating point shader core with multiple independent processors are designed to handle any type of shading operations, including vertex shading and pixel shading.” Ex. 47, Foundations of 3D Graphics Programming (https://www.google.com/books/edition/Foundations_of_3D_Graphics_Programming/UJIpzmCpPXcC?hl=en&gbpv=1 (2008), p.264).

72. The phrase “shader core” is rooted in an evolutionary change in the design of GPUs to what is now known as a “unified shader” based design. The industry, including NVIDIA, credit ATI with being first in the world to implement a unified shader based GPU in the Xenos chip that ATI designed for the Xbox 360 game console. See Ex. 38, NVIDIA, D. Luebke and G. Humphreys, “How GPUs Work,” in *Computer*, vol. 40, no. 2, pp. 96-100 at 99, Feb. 2007, doi: 10.1109/MC.2007.59. As detailed in an article written by a computer graphics industry forum, known as Beyond3D, the “main graphics chip” in ATI’s design “could be considered as a ‘shader core’ as this is one of its primary tasks. Beyond3D at 4. After ATI introduced the Xenos, NVIDIA introduced its own unified shader based GPU known as the “GeForce 8800” chip. NVIDIA paper at 99. In its technical briefings related to the GeForce 8800, NVIDIA used the phrase “shader core” to refer to the portion of their graphics processing unit circuitry to which “[d]ata coming in the top left of the unified design (such as vertices), are dispatched to the shader core for processing, and results are sent back to the top of the shader core, where they are dispatched again, processed again, looped to the top, and so on until all shader operations are performed and the pixel fragment is passed on to the [raster operations] subsystem.”

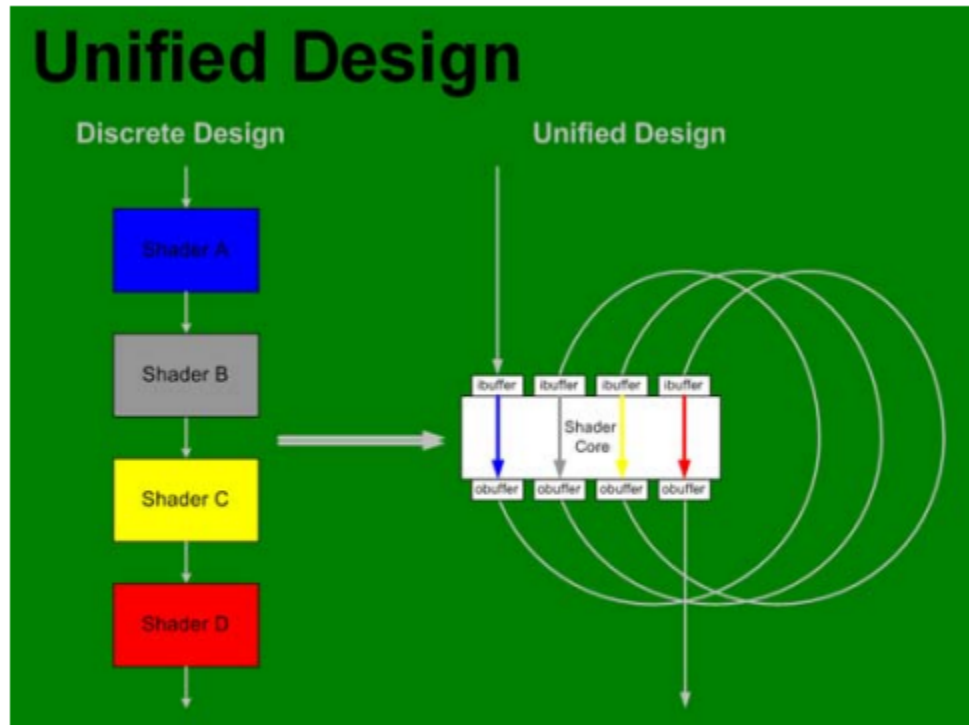


Figure 13. Classic vs. Unified Shader Architecture

NVIDIA, Technical Brief, NVIDIA GeForce 8800 GPU Architecture Overview (November 2006), 21. As further detailed by NVIDIA, the GeForce 8800 consisted of eight such “shader cores[,]” any of which “could be assigned to [a given] shader task[.]” Ex. 39, P. Glaskowsky, “NVIDIA’s Fermi: The First Complete GPU Computing Architecture,” Sep. 2009, 14.

73. Respondents ignore the term “shader” and its context of graphics processing in their proposed construction of “shader core.” A POSITA would have understood the term “shader” from its original use in the Hanrahan and Lawson paper. Ex. 35, “A Language for Shading and Lighting Calculations,” Proc. SIGGRAPH 90. “A shading language provides a means to extend the shading and lighting formulae used by a rendering system.” Hanrahan at Abstract. One of the three goals of a “shader” was to “provide a high-level language which is easy to use. It should have features - point and color types and operators, integration statements, built-in functions - that allow shading

calculations to be expressed naturally and succinctly.” Hanrahan at p. 290. A POSITA would have understood that a “shader core” was designed to run “shaders programs (e.g. geometry shader, vertex shader, pixel shader, tessellation shader, or the like) to complete a single graphics-processing task” ‘381 Patent at 8:36-38, *see also Id.* at 1:47-57. As shown in Fig. 2 and described by the ‘053 Patent (“Section III. An Example GPU” 1:40-9:8), a shader core is a component of “a complex integrated circuit that is adapted to perform data-parallel computing tasks, such as graphics-processing tasks.” ‘381 Patent at 1:23-25. For example, Fig. 2 includes a scan converter designed to recirculate the results of a shader core’s execution of a vertex shader program, to convert vertices into pixels for further processing by the shader core execution of a pixel shader program. The word “shader” in the term “shader core” means that the “shader core” is that part of the processor that processes graphics.

74. A POSITA would understand that the elements of claim 15 of the ‘381 Patent are circuitry or logic hardware. In addition, the fact that different claimed components, for example, the shader core are coupled to other circuitry indicates to the POSITA that the shader core is also circuitry.

75. Respondent’s construction incorporates the term “array” which does not appear anywhere in the ‘053 Patent. A POSITA would have understood a “SIMD array processor” to consist of a plurality of processors each running the same instruction, but each operating on different data. Hence a SIMD array processor would execute an instruction on an entire array of data instead of just a single data element. A POSITA would have understood a shader core to comprise a plurality of processing elements, but would also have understood that each and every one of the processing elements in a shader core need not execute the same instruction at the same time. For example, a POSITA would have understood that a shader core could comprise two or

more independent SIMD array processors.

D. “task”

76. A POSITA would have understood the term “task” means a “unit of processing work, comprising one or more threads and/or shader programs, as provided by the scheduling module to an engine.” as proposed by AMD.

77. I understand that Respondents have proposed that “task” be construed to mean “application, or portion thereof, sent by the CPU that includes a priority.” I disagree with their construction.

78. A POSITA would have understood from the ‘381 Patent at 1:47-2:7 that a “task” includes a “plurality of ... threads.”

79. A POSITA would understand the term “task” to have its plain meaning. The following definitions are consistent with a POSITA’s understanding of the term “task”:

- “task: a piece of assigned work.” Ex. 40, The American Heritage Dictionary (4th ed)
- “task (1) (A) (software) A sequence of instructions treated as a basic unit of work by the supervisory program of an operating system.” Ex. 41, The Authoritative Dictionary of IEEE Standards Terms (IEEE 100) (7th ed.)
- “task: An executing process that, in a multitasking operating system, can appear to run simultaneously with other tasks.” Ex. 42, Webster's New World Computer Dictionary (10th ed.)

80. According to the ‘381 Patent, claim 15, tasks are “receive[d] from the scheduling module associated with a second processing unit.” Based on the claim language, a POSITA would conclude that the scheduler provides the unit of processing work that is a task.

81. The specification teaches that “In a conventional system, an OS scheduler would provide tasks 404 to a single command buffer in a serial fashion, and a conventional GPU would serially process the tasks. Unlike such a conventional system, OS scheduler 310 provides each task 404 to one of a plurality of command buffers 420A-N based on the priority type specified by the application 402.” ‘381 Patent at 10:22-25. Those commands are provided to “a plurality of virtual engines 432A-N. ‘381 Patent at 10:33-39. The specification then provides that “Tasks from virtual engines 432 are then (substantially) simultaneously executed by shader core 214 as described above.” ‘381 Patent at 10:39-41. Based on that disclosure, a POSITA would understand that the claimed “scheduler” is the OS scheduler 310 described by the ‘381 Patent. ‘381 Patent at 10:22-41. A POSITA would also understand that the scheduler provides the unit of processing work that is a task to the shader core.

E. “engine” / “plurality of engines”

82. A POSITA would have understood the term “engine” to mean “portion of the first processing unit circuitry, configured to cause one or more tasks to be independently executed on a shader core that is shared with other engines” as proposed by AMD.

83. I understand that Respondents have proposed a construction to the longer phrase of “a plurality of engines” as “two or more virtual modules that are each associated with a particular task received from the operating system.” I disagree with their construction.

84. A POSITA would understand that the term “first processing unit” as it appears in claim 15 (“the first processing unit comprises...a plurality of engines.”) would be circuitry, not software. The disclosure of the ‘381 Patent, structure of the claim, and the fact that the first processing unit, bus, and memory are coupled together would lead a POSITA to conclude that the first processing unit is structural and includes circuitry structure to, the claimed engine.

85. A POSITA would understand the disclosure of the ‘381 Patent as describing

hardware circuitry. The first processing unit is described as a GPU, which is “a complex integrated circuit that is adapted to perform data-parallel computing tasks, such as graphics-processing tasks.” ‘381 Patent at 1:23-25. The specification describes the possible engines as “a first background engine 202A, a second background engine 202B, a real-time low-latency engine 202C, a primary 3D engine 202D, and a low-latency 3D engine 202E.” ‘381 Patent at 7:5-8. Those engines are part of a “command processor” portion of that GPU. *Id.* Therefore the engines are hardware circuitry. In addition, the command processor circuitry, including the engines incorporated therein, are coupled to other circuitry within the GPU, including FIFOs 204A and 204B, Arbitration Logic 222, and Memory System 210. *See* ‘381 Patent at Fig. 2. A POSITA would understand that in order to be coupled to other circuit elements within a GPU embodied in an integrated circuit, the command processor and engines would be circuitry.

86. Given the disclosure of the ‘381 Patent, a POSITA would understand the term “virtual” to refer to a function of the engines, not their structure. Specifically, the ‘381 Patent specification discloses that the “[a]s mentioned above, GPU 110 includes a plurality of virtual engines embodied on a shader core.” ‘381 Patent at 6:42-43. The specification also provides that “[a] GPU is a complex integrated circuit that is adapted to perform data-parallel computing tasks, such as graphics-processing tasks.” ‘381 Patent at 1:23-25.

87. A POSITA would understand that the term “virtual” meant to describe that by enabling substantially simultaneous execution of tasks on a single shared shader core, the engines function to virtualize the single shared shader core as a resource from the viewpoint of the outside world, thereby creating the appearance that the GPU actually contains multiple shader cores.

88. According to the ‘381 Patent, a POSITA would understand that the groupings of tasks is a design choice. Even though the ‘381 Patent engines are designed to handle particular

types of tasks (e.g., “a first background engine 202A, a second background engine 202B, a real-time low-latency engine 202C, a primary 3D engine 202D, and a low-latency 3D engine 202E”), those engines could be designed to group tasks in a different manner.

F. “load”

89. A POSITA would have understood the term “load” to have its plain and ordinary meaning.

90. A POSITA would understand the term “task” to have its plain meaning. The following definitions are consistent with a POSITA’s understanding of the term “task”:

- “put . . . a program in (a computer)” Ex. 43, Oxford Dictionary of Current English (1996);
- “transfer (data) from storage into a computer’s memory” Ex. 40, American Heritage Dictionary (2001);
- “transfer (a program or data) into memo, or into the central processor from storage” Ex. 44, Concise Oxford American Dictionary (2006);
- “to transfer (data) from a storage device into a computer’s memory” Ex. 46, American Heritage College Dictionary (2007);
- “enter data or programs into storage or working registers,” or “to read machine code into main memory in preparation for execution” Ex. 41, IEEE (2000); and
- “place information from storage into memory for processing” Ex. 45, Microsoft Computer Dictionary (2002).

91. The intrinsic record of the ’381 Patent uses load in a manner consistent with this plain meaning.