

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. QUALIFICATIONS AND EXPERIENCE.....	2
III. MATERIALS CONSIDERED	4
IV. LEGAL STANDARDS	5
A. Claim Construction.....	5
B. Obviousness	6
V. CLAIM CONSTRUCTION	8
VI. RELEVANT TECHNICAL BACKGROUND AT THE TIME OF THE '381 PATENT	10
A. Basic Graphics Processing.....	10
B. Tasks Versus Threads	11
C. Multi-Threading Versus Simultaneous Multi-Tasking	13
D. Shader Programs.....	14
VII. '381 PATENT OVERVIEW	15
A. Parallel Engines	18
B. Engines Versus Shader Programs.....	19
C. Simultaneous Multi-Tasking	21
VIII. DESCRIPTION OF ASSERTED ART	22
A. Wilt (U.S. Patent No. 8,151,095)	22
B. Jiao (U.S. Patent No. 8,319,774)	25
C. Stuttard (International Publication No. WO 00/62182)	29
IX. EVALUATION OF PETITIONER'S PROPOSED GROUNDS	31
A. Ground 1: Wilt and the Knowledge of a POSITA Do Not Render Claims 15-20 of the '381 Patent Obvious.....	31
1. Wilt Fails To Disclose or Render Obvious the Claimed "Plurality of Engines".....	32
2. Wilt Fails To Disclose or Render Obvious the Claimed "Shader Core" ..	40

B.	Ground 2: Jiao and the Knowledge of A POSITA Do Not Render Claims 15-20 of the '381 Patent Obvious.....	43
1.	Jiao Fails To Disclose or Render Obvious the Claimed “Plurality Of Engines”.....	43
2.	Jiao Fails To Disclose or Render Obvious the Claimed “Scheduling Module” and “Second Processing Unit” Because the Alleged Scheduling Modules Are Not Associated With the Alleged Second Processing Units	63
3.	Jiao Fails To Disclose or Render Obvious the Claimed “Shader Core” ...	66
C.	Ground 3: Stuttard and the Knowledge of A POSITA Do Not Render Claims 15-20 of the '381 Patent Obvious	74
1.	Stuttard Fails to Disclose or Render Obvious the Claimed “Shader Core” Because the Alleged Shader Core Does Not Execute a First Task While Executing a Second Task.....	74
2.	Stuttard Fails to Disclose or Render Obvious the Claimed “Scheduling Module” and “Second Processing Unit”	79
3.	Stuttard Fails to Disclose or Render Obvious the Claimed “Plurality of Engines”.....	85
X.	CONCLUSION.....	89

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

I. INTRODUCTION

1. I have been retained by ATI Technologies UCL (“ATI” or “Patent Owner”), as an independent expert in this proceeding before the Patent Trial and Appeal Board (“PTAB” or “Board”). I understand that Realtek is requesting that the Board institute an Inter Partes Review (“IPR”) proceeding of U.S. Patent No. 8,854,381 (the “’381 patent”) (Ex. 1001) in IPR2023-00687.

2. I am providing this testimony for consideration by the Patent Trial and Appeal Board in this proceeding. Many of my opinions are also responsive to the Declaration of Massoud Pedram (“Pedram Decl.”) (Ex. 1003), which I understand was submitted in support of Realtek’s Petition for *Inter Partes* Review of the ’381 patent. I have personal knowledge of the facts and opinions set forth in this declaration and all of the opinions and conclusions found in this declaration are my own.

3. I am being compensated at my standard hourly rate for my services. I have no personal or financial stake or interest in the outcome of the present proceedings, and my compensation is not dependent in any way upon the outcome of this proceeding.

II. QUALIFICATIONS AND EXPERIENCE

4. I have been a professor of computer science, focusing on 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 Hawthorne, 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 served as the founding computer graphics area editor of the ACM Books series, and oversaw the 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 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. I am being compensated for my time working on this matter at my usual rate of \$575 per hour. My compensation does not depend on the outcome of this litigation.

12. I believe I meet the criteria for a POSITA as laid out in Investigation No. 337-TA-1318, Ex. 1008 at 9, and any other appropriate definition of a POSITA for this patent.

13. Additional details about my employment history, fields of expertise, and publications are further described in my curriculum vitae, which is attached as Exhibit A to this report.

III. MATERIALS CONSIDERED

14. In forming my opinion I have relied on the following documents:

Exhibit	Description
EX1001	U.S. Patent 8,854,381 to Lefebvre et al.
EX1002	Prosecution History of U.S. Patent 8,854,381
EX1003	Declaration of Massoud Pedram, Ph.D.
EX1004	U.S. Patent No. 8,151,095 to Wilt (“Wilt”)
EX1005	U.S. Patent No. 8,319,774 to Jiao <i>et al.</i> (“Jiao”)

Exhibit	Description
EX1006	PCT Pub. No. WO00/62182A2 to Stuttard <i>et al.</i> (“Stuttard”)
EX1007	Curriculum Vitae of Massoud Pedram, Ph.D.
EX1008	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318 (Order No. 31, Oct. 18, 2022)
2002	Pat Hanrahan and Jim Lawson. 1990 “A Language for Shading and Lighting Calculations.” Proc. SIGGRAPH ’90. pp. 289-298
2003	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318, Final Initial Determination (July 7, 2023)
2004	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318, Respondents Post-Hearing Brief (April 10, 2023) (Redacted)
2005	<i>Certain Graphics Systems, Components Thereof, and Digital Televisions Containing The Same</i> , Inv. No. 337-TA-1318, Deposition Transcript of Dr. Pedram (January 4, 2023)

IV. LEGAL STANDARDS

15. In preparing and expressing my opinions and considering the subject matter of the ’381 patent, I am relying on certain basic legal principles that counsel have explained to me. These principles are discussed below.

A. Claim Construction

16. I understand that the first step in determining the validity of a claim is for the claim to be properly construed. I have been further advised that, in an *inter*

partes review proceeding, the claims of a patent are typically given their ordinary and customary meaning as would be understood by one of ordinary skill in the art in the context of the claims and the patent disclosure (specification).

17. I further understand that an inventor can provide special definitions for specific claim term(s) within the patent. However, any special definition for a claim term must be set forth in the specification with reasonable clarity and precision.

B. Obviousness

18. I have been informed that a claim may be invalid as obvious if the subject matter described by the claim as a whole would have been obvious to a POSITA at the time the claimed invention was made. I understand that the standard for obviousness in an *inter partes* review proceeding is by a preponderance of the evidence.

19. I have also been informed that a determination of obviousness involves an analysis of the scope and content of the prior art, the similarities between the claimed invention and the prior art, and the level of ordinary skill in the art. I have been informed and understand that a prior art reference should be viewed as a whole.

20. I have been informed that in considering whether an invention for a claimed combination would have been obvious, I may assess whether there are apparent reasons to combine known elements in the prior art in the manner claimed in view of interrelated teachings of multiple prior art references, the effects of

demands known to the design community or present in the market place, and/or the background knowledge possessed by a POSITA. I also understand that other principles may be relied on in evaluating whether a claimed invention would have been obvious.

21. I have been informed that, in making a determination as to whether or not the claimed invention would have been obvious to a POSITA, one may consider certain objective indicators of non-obviousness if they are present, such as: commercial success of product(s) practicing the claimed invention; long-felt but unsolved need; teaching away; unexpected results; copying; and praise by others in the field. I understand that for such objective evidence to be relevant to the non-obviousness of a claim, however, there must be a causal relationship (called a “nexus”) between the claim and the evidence. I also understand that this nexus must be based on a novel element of the claim rather than something available in the prior art.

22. I have also been informed and understand that when considering the obviousness of a patent claim, one should consider whether a reason or motivation existed for combining the elements of the references in the manner claimed, and that the prior art must create a reasonable expectation of success in producing the claimed subject matter.

V. CLAIM CONSTRUCTION

23. I understand certain terms of the '381 patent have been previously construed by the ALJ in the ITC Investigation. *See* Ex. 1008 at 36-41. Dr. Pedram applied the ALJ's constructions for the terms "shader core," "task," and "load." Ex. 1003, ¶¶ 40, 41, 44. I have applied these constructions as well.

Term	Construction
"shader core"	"[A]n array of processing elements." Ex. 1008 at 33-36.
"task"	"[A] unit of processing work or portion of an application to be executed," and it need not include a priority. Ex. 1008 at 36- 38.
"load"	Construed not to require extracting and storing. Ex. 1008 at 41-41.

24. With respect to the term "task," Dr. Pedram has opined that "threads are not tasks." Ex. 1003, ¶ 102. I have also applied that construction to the term "task." Furthermore, as explained below in Sections VI.B and VI.C, I agree with this additional clarification because a task is a higher-level item. As a single task is comprised of a plurality of threads, it is not possible to conclude from a reference's mere disclosure of a plurality of threads whether those threads correspond to a single task or multiple tasks.

25. Dr. Pedram also argues for constructions of the terms "execute a first task . . . while executing a second task" and "engine" / "plurality of engines." I have been asked to apply the constructions applied by Dr. Pedram.

Term	Construction
“execute a first task ... while executing a second task”	“the first task is executed at the same time as the second task, <i>i.e.</i> , the tasks are executed simultaneously” Ex. 1003, ¶ 42.
“engine”/“plurality of engines”	“not limiting” “to either hardware or software.” Ex. 1003, ¶ 43.

26. Additionally, Dr. Pedram appears to take the following constructions for the terms “each of the plurality of tasks,” and “configured to.” I have been asked to apply these constructions as well:

- As to the term “**each of the plurality of tasks**”: Petitioner “assert[ed] that each individual engine ‘must be configured to load state data for each of the pluralities of tasks that are received.’” Ex. 2003 (ID) at 99-100; Ex. 2004 (Pet. Redacted Post-Hearing Br.) at 73-76. I agree that each engine must be configured to load state data, although I do not agree that each item of state data must correspond to a plurality of tasks, *i.e.*, more than one task.
- As to the term “**configured to**”: Petitioner contended that the claim language requires “the design of an apparatus to perform an action and not merely for the action to occur some of the time.” Ex. 2003 (ID) at 97-98; Ex. 2004 (Pet. Redacted Post-Hearing Br.) at 67-68. Further, as Petitioner argued in the ITC, because the claim requires the system to be “configured to” execute a graphic task while executing a general

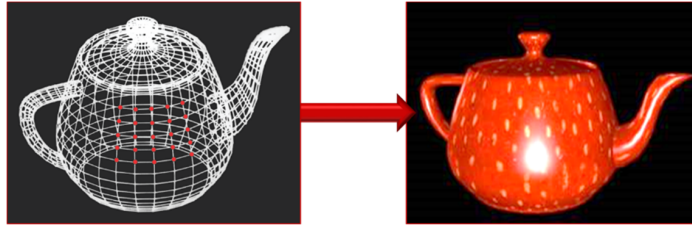
compute task, it is insufficient to point to “the result of chance or probability, which only requires capability.” *Id.* at 67. As Petitioner argued there, “configured to” refers to the system’s “design . . . not its use at any given point in time.” *Id.* Petitioner argued that “the shader core must be specifically programmed or designed to cause a first task and a second task to be executed simultaneously—and not just any two tasks, but a first task and a second task from the same plurality of tasks received by the plurality of engines. If a first task and a second task from a plurality of tasks are occasionally executed at the same time, and it happens as a matter of probability or chance, this is nothing more than a shader core that is capable of executing a first task and a second task simultaneously.” *Id.* at 67-68.

VI. RELEVANT TECHNICAL BACKGROUND AT THE TIME OF THE '381 PATENT

A. Basic Graphics Processing

27. Graphics processing involves manipulating and rendering 3D graphical objects onto a 2D display screen. Specifically, 3D objects (typically represented by a collection of “primitives” geometrical shapes such as triangles) are converted into a 2D image (typically comprising a collection of pixels) as shown below. The 2D image represents a depiction of the 3D objects, rendered from the perspective of a

given viewer, and shaded to convey the object's surface material illuminated from a given light source.



28. The conversion of a 3D shape consisting of a mesh of triangles into a photorealistic 2D image consisting of an array of pixels happens in a number of processing stages. The “pixel processing” stage relevant here includes the computation of the colors of each of these pixels/fragments, as well as a process called “texture mapping.”

B. Tasks Versus Threads

29. As Dr. Pedram opines, “[t]hreads are not tasks.” Ex. 1003, ¶ 102. Graphics software programs can create “tasks,” which are separated into multiple sequences of instructions, called threads, to improve efficiency. The ’381 patent explains that

[t]o 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, such as a geometry shader, a pixel shader, a vertex shader, or the like. Each thread (e.g., shader program) is typically associated with a set of state data (such as texture handles, shader constants, transform matrices, or the like) that is locally stored in data-storage units of the GPU. The locally stored state data is called a context.

Ex. 1001 at 1:49-57.

30. A task is a higher-level item than a single instruction or a thread, and a thread (sequence of instructions) is a higher-level item than an instruction. *Id.* An exemplary “task” is, in an example that Dr. Pedram gives, “drawing a triangle of given shape and size with white boundaries filled with a solid blue color in the right upper corner of a 1920 x 1080 LCD Display.” Ex. 1003, ¶ 102. One example “thread” for Dr. Pedram’s exemplary task, would be for a vertex shader program to convert a location it receives representing one vertex point, from 3D space to 2D space. For a given task, each of the vertex shader, geometry shader, and pixel shader would have many, many threads. For example, the pixel shader would have one thread for each of the operations it executes on each individual pixel. Each thread reveals nothing about the “task” from which it was decomposed, because each thread is such a small portion, bearing no discernable relationship to the task. For example, the thread that translates one vertex from 3D space to 2D space in and of itself does not enable one to identify the full exemplary task above.

31. Even taking multiple threads together, a POSITA would not know what the ultimate “task” from which the threads were decomposed was, much like knowing a few letters that were once part of a sentence does not enable one to reconstruct the original sentence. Moreover, multiple threads executing at any given time may not even be related to the same task.

C. Multi-Threading Versus Simultaneous Multi-Tasking

32. An ordinary “single-threaded” graphics processing system executes a single thread, proceeding instruction by instruction, in their original order, on one piece of data at a time. A multi-threaded processing system improves upon that in a few ways. First, using an architecture called “single-instruction, multiple data,” a single instruction can be received by multiple processing units that apply that instruction to different data at the same time. Ex. 1001 at 1:58-2:5. Notably, this system does not execute multiple *different* instructions or threads simultaneously; it is the same one instruction at any given time. The Stuttard reference, for example, is one of these multithreaded processing systems that can only execute one instruction at a time, as explained below in Section IX.C.1. Second, some multi-threaded systems can execute instructions in a different order from how they were received. Ex. 1001 at 1:58-2:37. Again, although this system improves the single-threaded architecture, it does not execute multiple instructions simultaneously. *Id.*

33. Such multi-threaded systems have substantial disadvantages, though. First, switching is time-consuming. *Id.* at 2:38-43, 46-56. Second, it requires the system to have additional local memory to hold data for an instruction that is in the middle of execution at the time of switch, so that progress is not lost and the system can later switch back and keep executing without starting from the beginning. *Id.* at 2:43-46. As explained more fully below, the ’381 patent solves these problems using

a particular multi-engine architecture that shares processing resources of a “shader core” that can simultaneously execute multiple tasks, without the disadvantages of the prior art.

D. Shader Programs

34. Shader programs, such as vertex shaders, geometry shaders, and pixel shaders, are pre-programmed software modules that receive data and operate on them. The concept of a “shader” program dates back to Hanrahan and Lawson’s “A Language for Shading and Lighting Calculations” (Proc. SIGGRAPH ’90, pp. 289-298), which showed how Pixar’s non-real-time film production rendering system (which would take minutes if not hours to produce each frame) could include short custom programs called “shaders” that would be run pixel-by-pixel to compute the appearance of a surface. Ex. 2002. As the paper explains, these are “preprogrammed shaders” that are written in code, compiled once, loaded into the processor, and then run, and while running they process (or “evaluate”) data that they receive. Ex. 2002 at Sections 2 (“The shading language allows procedures, called shaders, to be written that compute the various terms in the above equation.”), 4 (describing the “implementation” of shader programs, starting with compilation of the code comprising the program, followed by loading the compiled program and the program evaluating data as it runs), Table 1 (showing only one compilation step).

35. Notably, as explained further below, shader programs are not intelligent processing units that can receive, interpret, and execute “tasks.” They do not receive “tasks” at all. Rather, a shader is a software program that is “call[ed],” as an ordinary software program is, to “to perform *its task*,” *i.e.*, the sole thing it was preprogrammed to do, on data it receives during runtime. *Id.* at Sections 3.3 (“Calling a shader to perform its task is under the control of the rendering system, and all information from the renderer is passed to the shader through external variables and not via its arguments”), 4.5 (“Eventually a geometric primitive has been reduced to *an object* for which the rendering system wishes to *invoke a shader*.”).

VII. '381 PATENT OVERVIEW

36. The '381 patent sought to solve the problems with prior-art graphics processing units (GPUs), including the issues with simpler multi-threaded architectures described above. *See* Section VI.C. The '381 patent introduced a set of engines, as shown, for example, as blocks 202A–E in Fig. 2 (reproduced below), that replaced this single-threaded and prior multi-threaded designs described above. Ex. 1001 at 4:30-5:45. This enabled the improved scheduling of tasks to the GPU to provide more tasks to the shader core. *Id.* Having more tasks available to the shader core means that the core is more likely to have a task available and ready for execution. *Id.* As the shader core grows to hundreds and thousands of individual processors, it becomes more challenging to produce enough tasks to keep them all

running at peak efficiency. *Id.* Further, the capability of each of these engines to load state data associated with their respective tasks serves to reduce latency relative to conventional systems which relied on a single functional unit or the CPU to load and switch between different state data (*e.g.*, context data). *Id.*

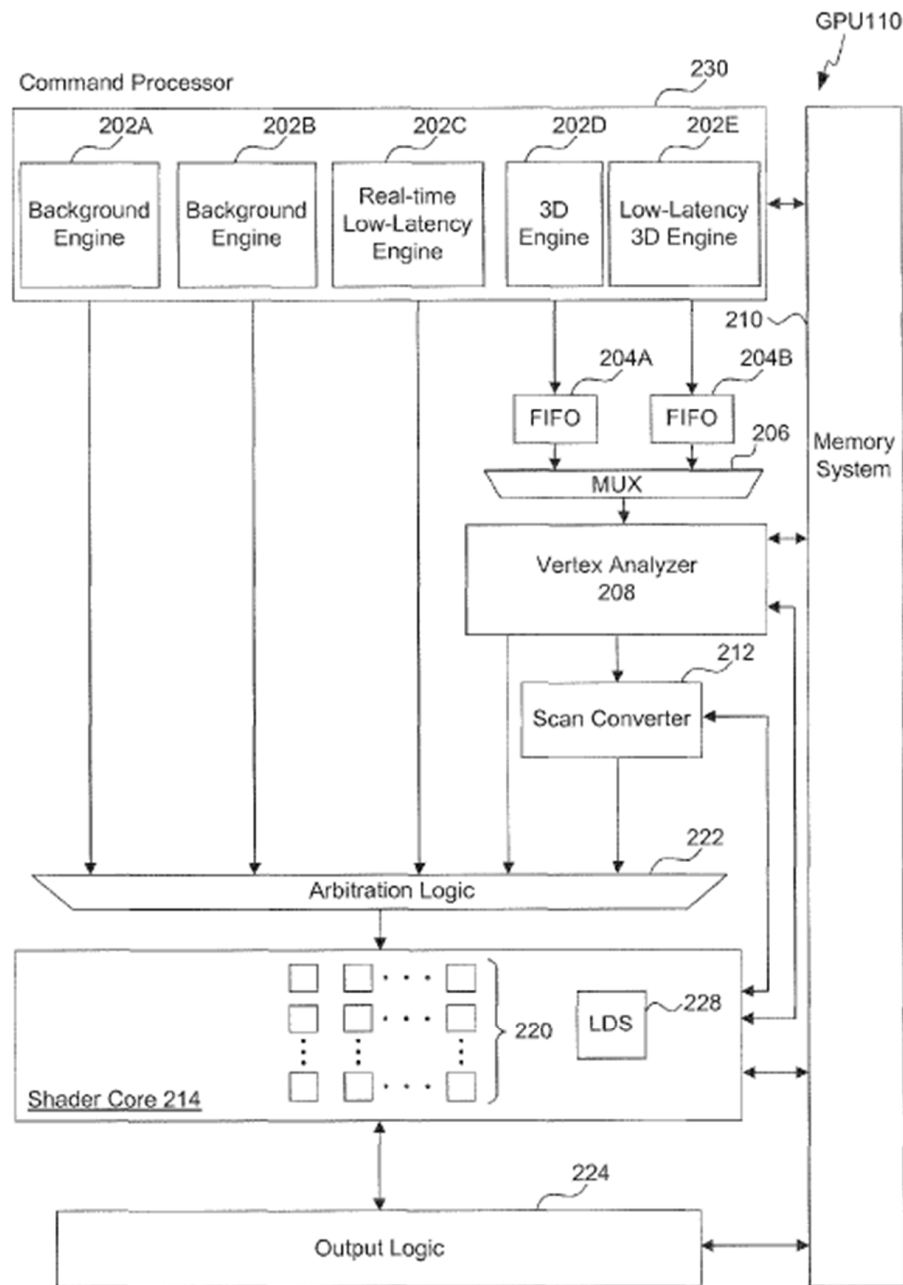


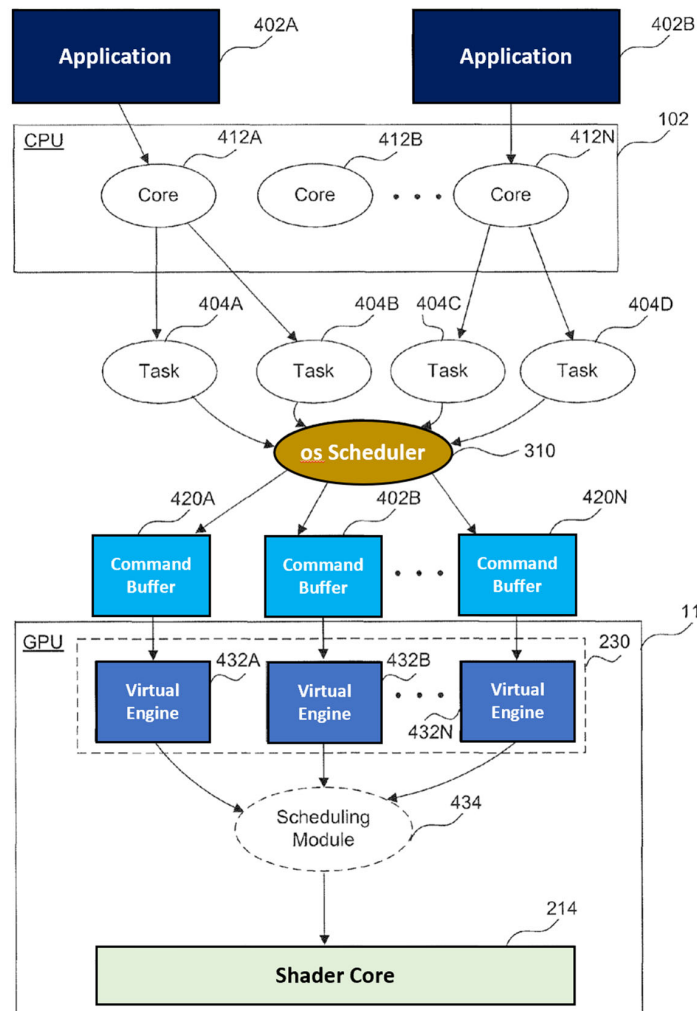
FIG. 2

Ex. 1001, Fig. 2. These engines are each configured to (i) receive, from a scheduling module (*e.g.*, an operating system (OS) scheduler), a plurality of tasks substantially

in parallel with each other and (ii) load a set of state data associated with each of the plurality of tasks. *Id.* at Abstract, 3:3-7.

A. Parallel Engines

37. The '381 patent distinguishes its parallel configuration of the engines from the conventional serial reception of tasks by the GPU as in the prior art. For example, the '381 patent explains that the engines function independently to receive tasks scheduled for execution by shared resources of the GPU. *Id.* at 4:48-51 (“Each virtual engine is configured to receive data-parallel processing tasks (e.g., graphics-processing tasks and general-compute tasks) and independently execute these tasks on the single shader core.”); *Id.* at 6:1-6 (“GPU 110 includes a plurality of virtual engines that share resources of a single shader core. In this way, the plurality of virtual engines of GPU 110 can execute a plurality of tasks substantially in parallel.”). This is depicted, for example, in Figure 4:



Ex. 1001 at Fig. 4 (annotated).

B. Engines Versus Shader Programs

38. Further, the engines described in the '381 patent are differentiated from other elements, including shader programs. For example, as shown in Fig. 2 (reproduced above on page 17) and as disclosed at 7:5-19, the graphics-processing tasks are received by, *e.g.*, the 3D engine 202D and the low-latency 3D engine 202E, which connect to certain graphics resources of the GPU, such as the vertex analyzer

208 and scan converter 212. The '381 patent describes that these graphics components may invoke shader programs to be run on the shader core: “Vertex analyzer 208 identifies shader programs associated with a graphics-processing and/or general-compute task and schedules when each shader program can be launched in shader core 214 based on input and output data that will be available.” Ex. 1001 at 7:42-46. “Processing elements 220 may implement a plurality of different shader programs (*e.g.*, geometry shader, vertex shader, pixel shader, tessellation shader, or the like).” *Id.* at 8:34-41. By this point in the processing chain, the engines 202 have already received the tasks, and as explained in the '381 patent and in Section VI.D above, the shader programs merely receive data and do not receive tasks: they are preprogrammed pieces of software that are invoked as part of carrying out a task, *i.e.*, they are constituent components of tasks. They bear no similarity to the “engines” recited in the '381 patent, nor do they perform any of the actions the engines are required to do in the challenged claims.

39. The '381 patent confirms in other places that “shader programs” are actually constituent components of tasks; they are not *engines*, which according to the claim language, receive tasks. The '381 patent teaches that in its terminology, “shader programs” are parts of “tasks”—they are not engines that *receive* tasks. More specifically, shader programs are threads, or parts of threads, which themselves are parts of tasks, as explained above in Section VI.B. The '381 patent

states that “[t]o 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, such as a geometry shader, a pixel shader, a vertex shader, or the like.*”). Ex. 1001 at 1:47-52. The ’381 patent further states, “[t]o efficiently execute the *various threads (e.g., shader programs)* of a single graphics-processing task, the GPU includes an array of processing elements, called a shader core.” *Id.* at 1:58-61.

C. Simultaneous Multi-Tasking

40. The ’381 patent also introduced the ability to run tasks at the same time by partitioning the shader core (element 214 in Fig. 2 above) to run separate tasks in parallel. The ’381 patent discloses that “[t]o enable shader core 214 to (substantially) simultaneously process tasks from the plurality of virtual engines of command processor 230, processing elements 220 of shader core 214 are partitioned in space and/or time.” *Id.* at 8:3-7. Further, the “spatial partitioning” of the shader core was implemented, in part, by organizing the shader core into multiple data-parallel subsets of processing elements.

41. By sharing available shader resources, the ’381 patent explains how the plurality of engines serve to improve utilization of the GPU relative to conventional approaches which included a single path to the GPU’s execution resources (e.g., the shader core). For example, the ’381 patent discloses that “[a]s illustrated in FIG. 3,

command processor 230 includes a plurality of virtual engines that share resources of GPU 110.” *Id.* at 6:67-7:2. Moreover, the ’381 patent discloses that “[s]haring the resources of the GPU to provide a plurality of virtual engines in accordance with embodiments of the present invention improves the use of the GPU resources, especially on large chips.” *Id.* at 5:29-32.

VIII. DESCRIPTION OF ASSERTED ART

A. Wilt (U.S. Patent No. 8,151,095)

42. Wilt (Ex. 1004) is directed to improvements in utilizing context data by processing units in a parallel processing subsystem. Specifically, Wilt explains that conventional systems may include multiple parallel processing units (PPUs) “that may be configured to beneficially provide a high volume of computational throughput that is impractical to achieve with a single processing unit.” Ex. 1004 (Wilt) at 1:11-14.

43. Figure 1 shows an overview of the system, with the parallel processing subsystem at issue here highlighted:

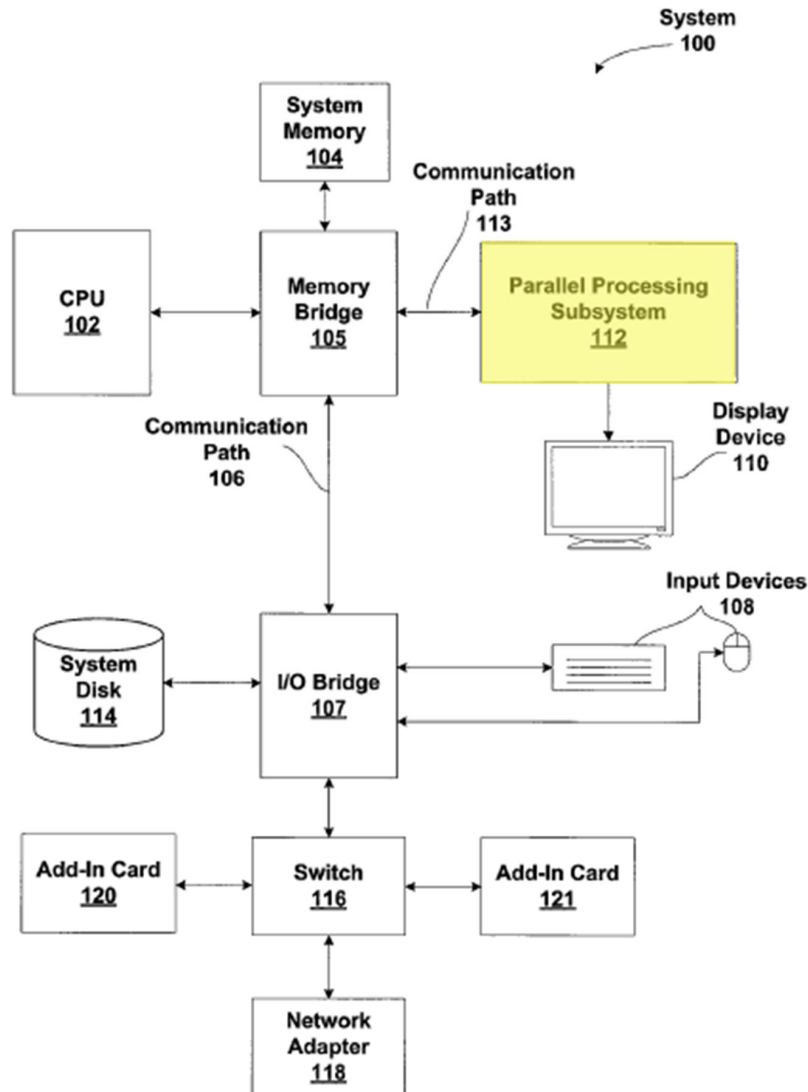


Figure 1

Ex. 1004, Fig. 1 (annotated). Figure 2 provides more detail on the parallel processing subsystem, including the multiple parallel processing units 220:

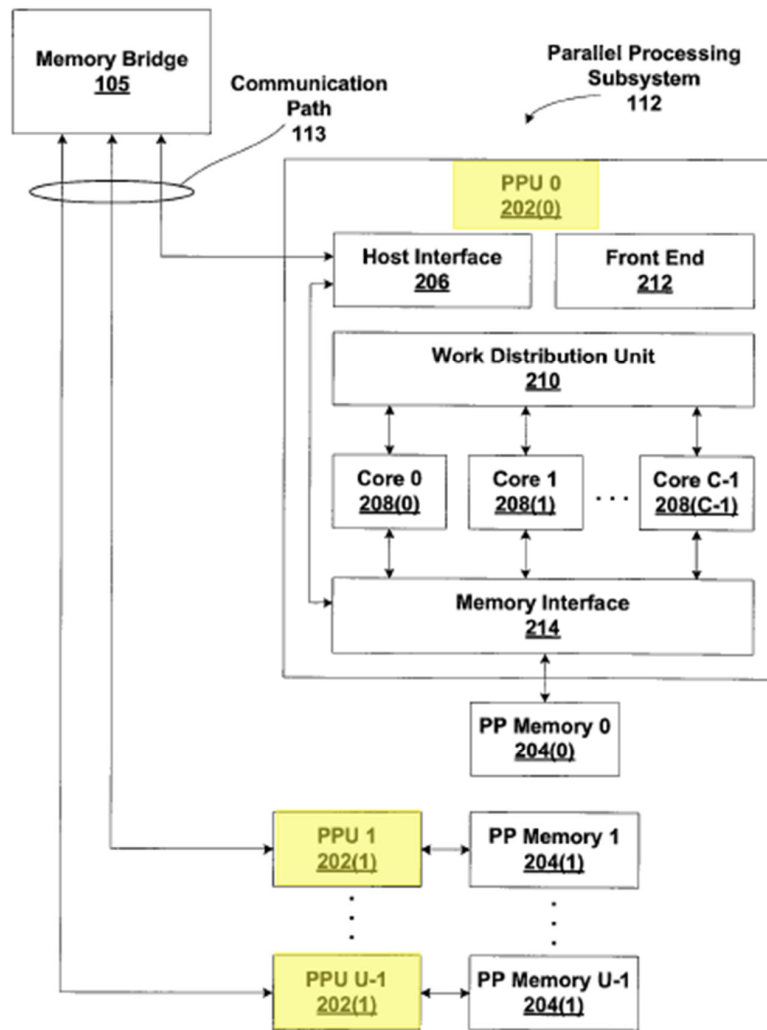


Figure 2

Ex. 1004, Fig. 2 (annotated). Figure 3 is further detail on the PPU 202:

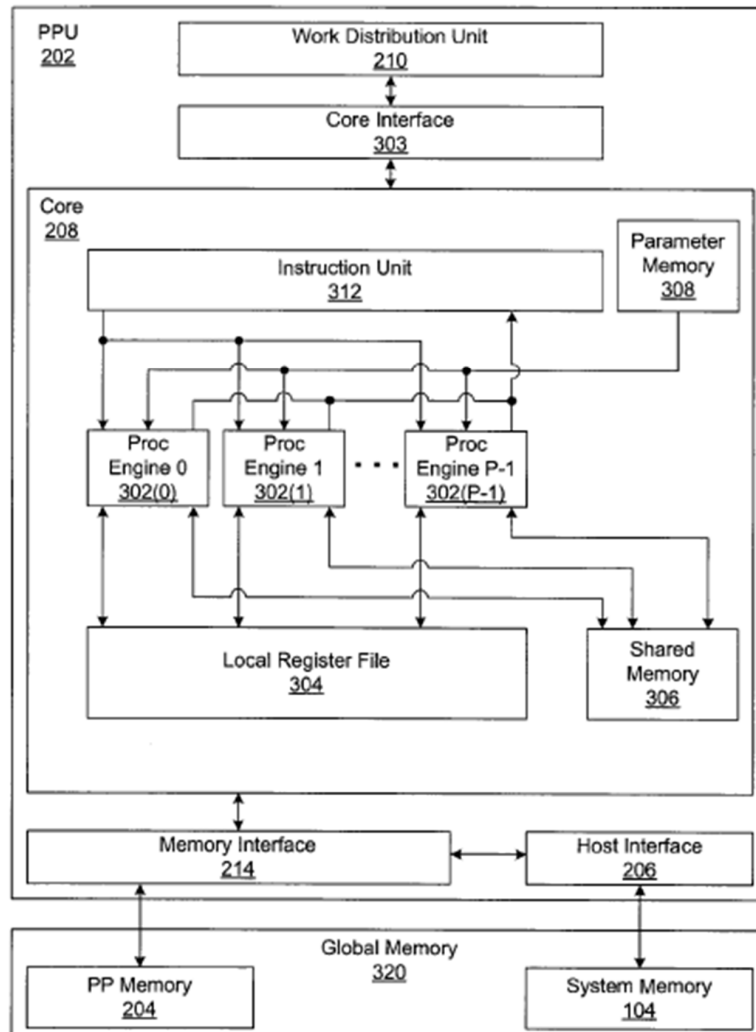


Figure 3

Ex. 1004, Fig. 3.

B. Jiao (U.S. Patent No. 8,319,774)

44. Jiao (Ex. 1005) is directed to a graphics processing system that is intended to address challenges in updating constants in a constant buffer, and the associated latencies encountered during a constant flush operation and the loading of new constant data.

45. Figure 1 of Jiao shows the GPU in the context of the larger architecture:

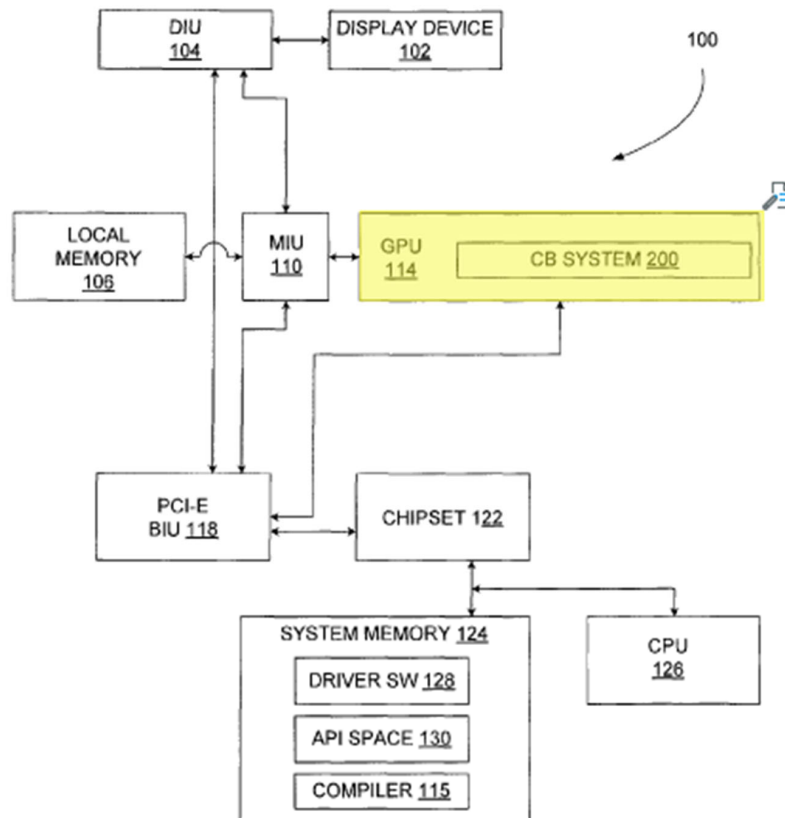


FIG. 1

Ex. 1005, Fig. 1 (annotated).

46. Figure 2 of Jiao is a high level diagram of the graphics processor 202 within GPU 114:

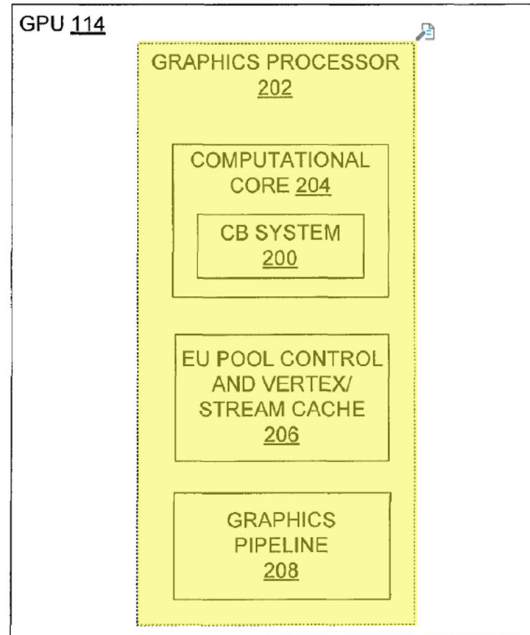


FIG. 2

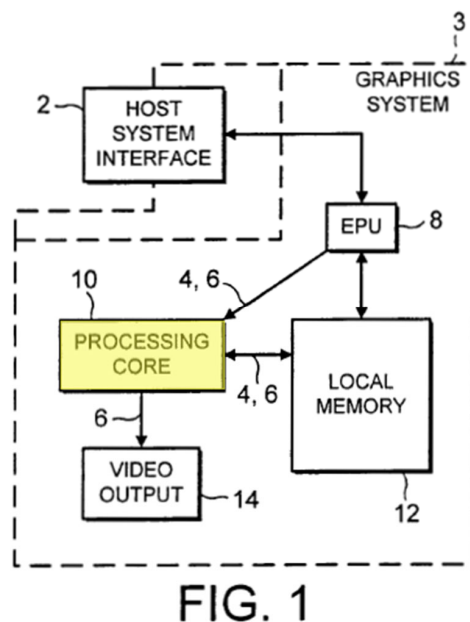
Ex. 1005, Fig. 2.

47. Figure 4 gives more detail of graphics processor 202. The key components are the Execution Units (EUs) 420 and EU Pool 414, the shader programs 304, 306, and 310, the EU Pool control 206, and the command stream processor 406.

C. Stuttard (International Publication No. WO 00/62182)

48. Stuttard (Ex. 1006) discloses an allegedly improved “single-instruction, multiple data” (“SIMD”) architecture that divides the processing elements into processing blocks. Ex. 1006 at 2:1-24.

49. Figure 1 gives an overview of the architecture of Stuttard:



Ex. 1006 at Fig. 1. The graphics system interfaces with a host system (not shown) via the host system interface 2. The graphics system 3 includes an embedded processing unit (EPU) 8 “for controlling the overall function of the graphics processor and for interfacing with the host system.” *Id.* at 4:26-30. Graphics processing takes place in the processing core 10. *Id.* at 7:25-27.

50. The processing core is shown in detail in Figure 3:

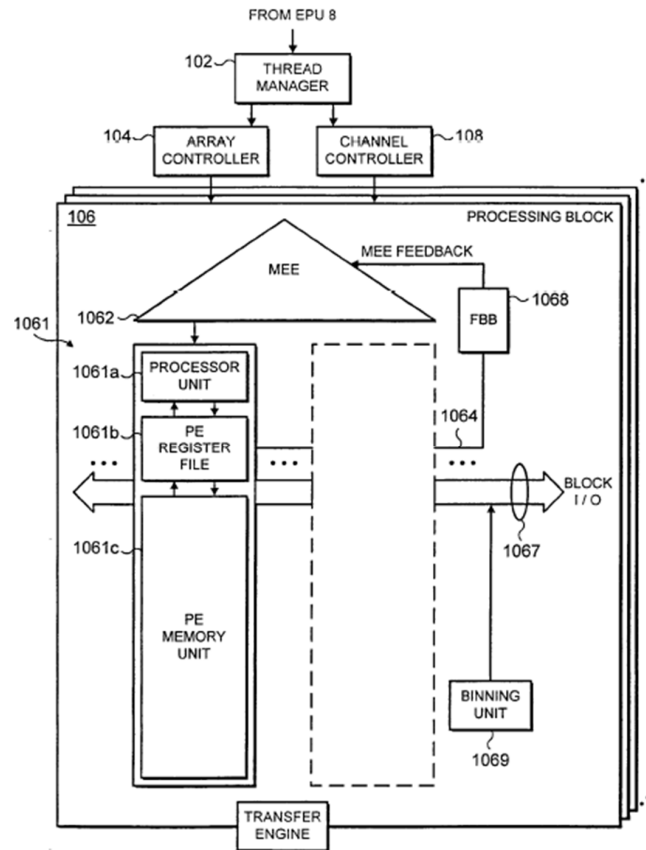


FIG. 3

Ex. 1006, Fig. 3. “The thread manager 102 is connected to receive control signals from the EPU 8. The control signals inform the thread manager as to when instructions are to be fetched and where the instructions are to be found. The thread manager 102 is connected to provide these instructions to the array controller 104 and to the channel controller 108. The array and channel controllers 104 and 108 are connected to transfer control signals to the processing blocks 106 dependent upon the received instructions.” *Id.* at 8:17-27. The processing core includes processing

blocks 106, which comprise an array 1061 of processor elements (PEs), a mathematical expression evaluator (MEE), and a binning unit. *Id.* at 8:15-9:3.

51. Critically, the entirety of Stuttard can only operate on one instruction at a time, and one “thread,” or “instruction stream” over a period of time, as explained more fully below with respect to Ground 3. Ex. 1006 at 4:17-23, 4:25-29, 10:20-24, 11:33-35; Pet. at 57 (referring to “thread scheduler 1025, which uses the status information to determine which *thread* should be active *at any one time*.”), 62 (“Stuttard’s thread scheduler recalculates which *thread* should be active”). All processing blocks 106 (Fig. 3) and all thread processors 1026 (Fig. 4) receive one instruction at a time out of the single instruction stream (thread) active over a given time period, and operate on different data, hence “single instruction, multiple data” (“SIMD”).

IX. EVALUATION OF PETITIONER’S PROPOSED GROUNDS

A. Ground 1: Wilt and the Knowledge of a POSITA Do Not Render Claims 15-20 of the ’381 Patent Obvious

52. In my opinion, claims 15-20 are not obvious in view of Wilt and the knowledge of a POSITA.

**1. Wilt Fails To Disclose or Render Obvious the Claimed
“Plurality of Engines”**

**a. The Alleged Engines Are Not Separate From the
Alleged Shader Core**

53. A POSITA would understand claim 15 to require “a plurality of engines configured to receive . . . a plurality of tasks” and “a shader core configured to *receive* the plurality of tasks *from* at least one of the plurality of engines.” Ex. 1001 at claim 15. The plain meaning of this language dictates that the plurality of engines cannot be inside the shader core, because if they were, the shader core would not receive the tasks from the engines. One item cannot receive a task from something within itself. Such an interpretation would also render redundant the separate “receive” steps, which require that the engines receive tasks and the shader core receive tasks. The ’381 patent specification confirms that the engines and the shader core are separate:

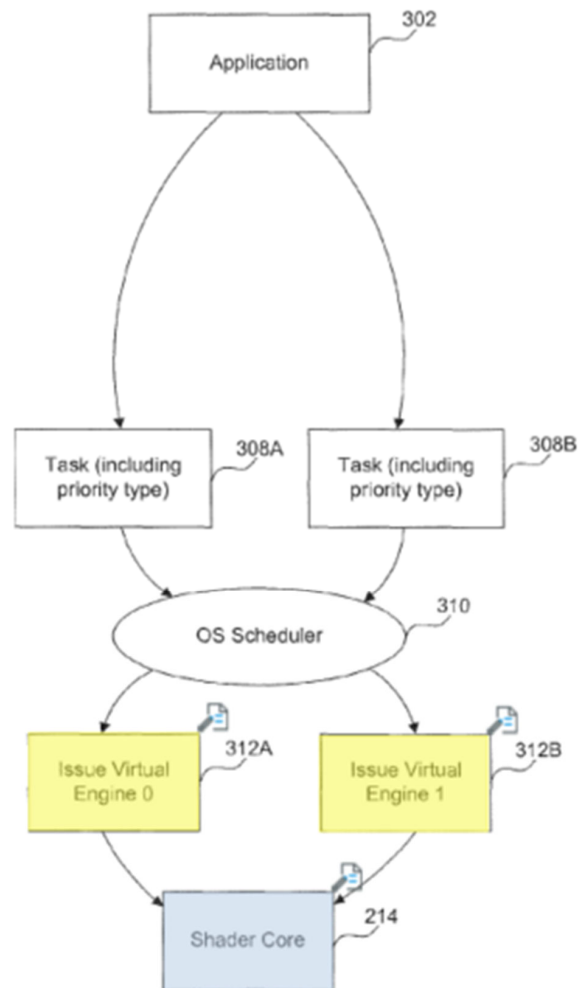


FIG. 3A

Ex. 1001 at Fig. 3A.

54. Dr. Pedram opined that Wilt’s work distribution unit 210, the front-end unit 212, and the core interface unit 303 are the “plurality of engines.” Ex. 1003, ¶ 71. I disagree, because these alleged “engines” are all within the alleged “shader

core.” As Dr. Pedram further opined, “[e]ach parallel processing unit (PPU) 202 is the claimed ‘shader core configured to receive the plurality of tasks.’” *Id.* at ¶ 76. Thus by the Dr. Pedram’s own admission, as depicted in Figures 2 and 3 below, each of the alleged engines, items 210, 212, and 303, is within the a alleged shader core, PPU 202.

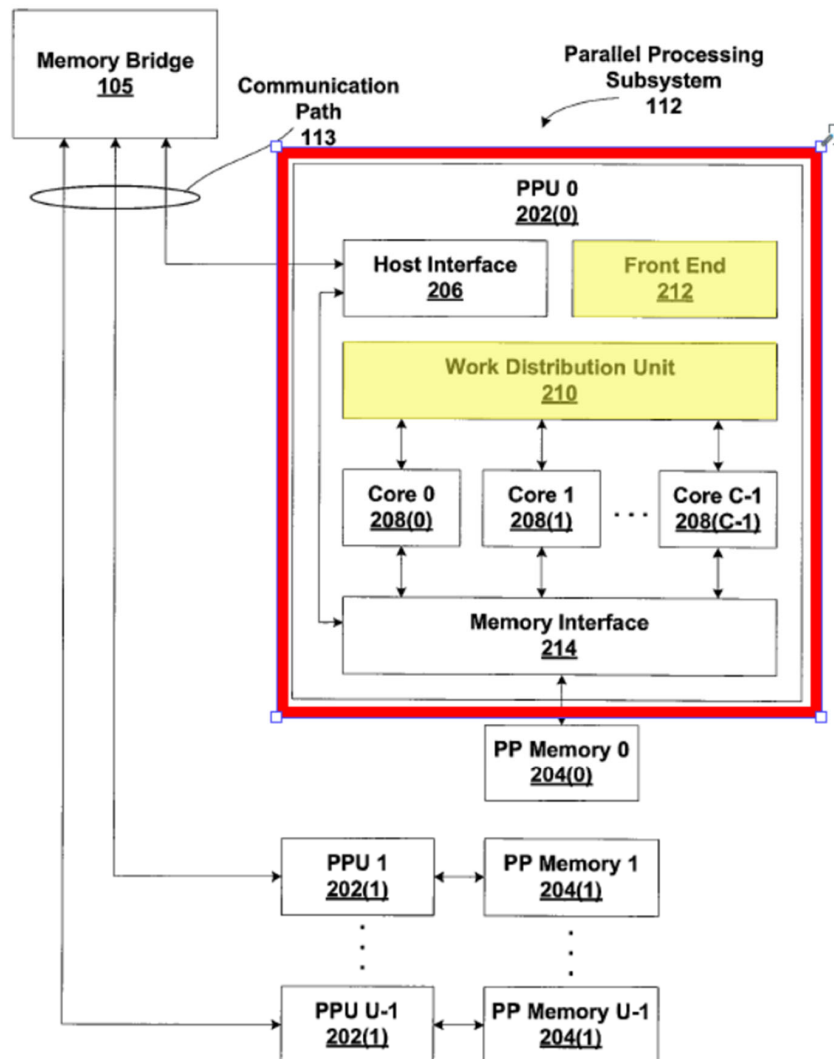


Figure 2

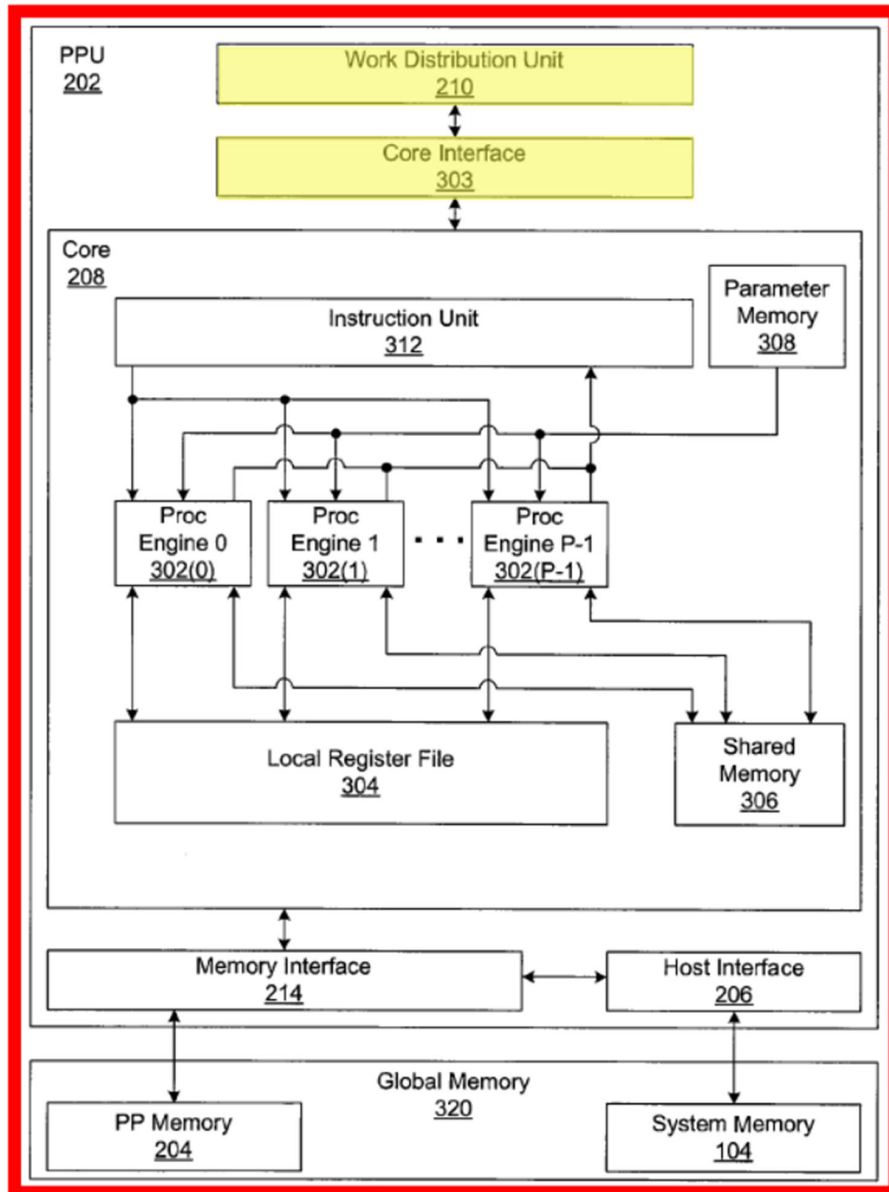


Figure 3

Ex. 1004 at Figs. 2-3 (annotated).

55. Dr. Pedram's alternative theory is also incorrect. Dr. Pedram states, "[a]lternatively, the work distribution unit 210, the front-end unit 212, and the core interface unit 303 of one PPU constitute one claimed 'engine,' and the engines of

the plurality PPU 202 constitute the claimed ‘plurality of engines.’” Ex. 1003, ¶ 73. For this theory, Dr. Pedram opines that “[a]lternatively, the plurality of parallel processing units (PPUs) 202 is the claimed ‘shader core configured to receive the plurality of tasks.’” *Id.* at ¶ 77. Under this theory, it remains true that each alleged engine (one instance of the combination of items 210, 212, and 303) is within the alleged shader core (the plurality of PPUs 202), and so this theory also fails.

b. The Alleged Engines Do Not Receive Tasks From The Alleged Scheduling Module

56. Claim 15 also requires the alleged engines receive tasks “from a scheduling module,” as required by claim 15. Ex. 1001 at Claim 15. However, Dr. Pedram argued that only the first item in a chain of items performs the “receiving” action—downstream items do not, because the reception has already been completed by the first item in the chain. The ’381 patent discloses the engines 202A as receiving tasks from the scheduling module. Ex. 1001 at Claim 15. Dr. Pedram was asked whether the downstream “vertex analyzer 208” and “scan converter 212” in Figure 2 of the ’381 patent *also* receive a task from the scheduling module, by virtue of the paths from engines 202 through FIFOs 204 and MUX 206:

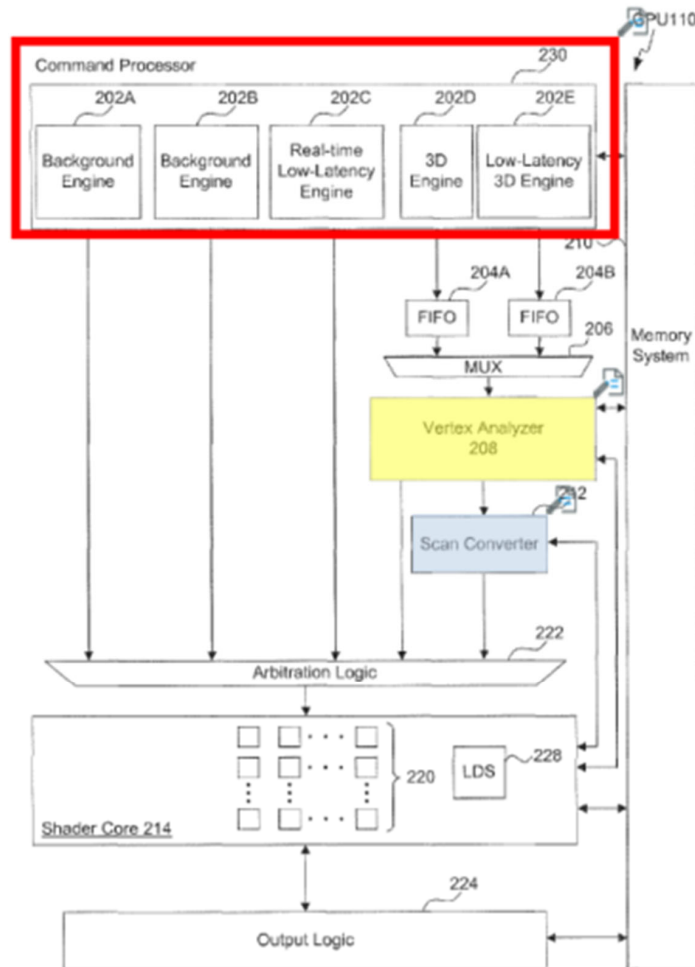


FIG. 2

Ex. 1001, Fig. 2 (annotated).

57. Dr. Pedram testified that vertex analyzer 208 and scan converter 212 do **not** receive tasks from a scheduling module. He testified that they receive tasks, but only from the engines 202A-202E at the top of Figure 2, not from a scheduling module. He testified that the engines are the sole units that receive the tasks from

the scheduling module, because once they receive such tasks, the action of receiving the task from a scheduling module has been completed. Ex. 2005 (Pedram 1/4/23 Dep. Tr.) at 84:9-88:12 (“So *vertex analyzer*, scan converter, they do receive tasks, but *they don’t receive it from the scheduling module*, they receive it from the engines. . . . [Q.] *what is the attribute of an engine that is -- that is missing from vertex analyzer* that disqualifies it from -- from being an engine? [A.] *it’s not receiving it from the scheduling module. The function has already been performed.*”). Dr. Pedram testified it would be “a ridiculous proposition” to argue that it is enough to merely be “coupl[ed]” in an indirect way to the scheduling module because “then everything that you see in [the] Figure 2, every component here you see . . . would be an engine.” *Id.* at 89:21-91:25.

58. For the same reason that the ’381 patent’s vertex analyzer 208 and scan converter 212 do not receive tasks from the alleged scheduling module, and only engines 202 do, neither do Wilt’s “work distribution unit 210, front-end unit 212, and core interface unit 303.” As depicted in Figure 2 of Wilt, to the extent anything in the processing unit receives a task from the alleged scheduling modules, which are the “software programs running on the host CPU,” it is the Host Interface 206, not any of the alleged engines, which are “the work distribution unit 210, the front-end unit 212, and the core interface unit 303.” Pet. at 26; Ex. 1004.

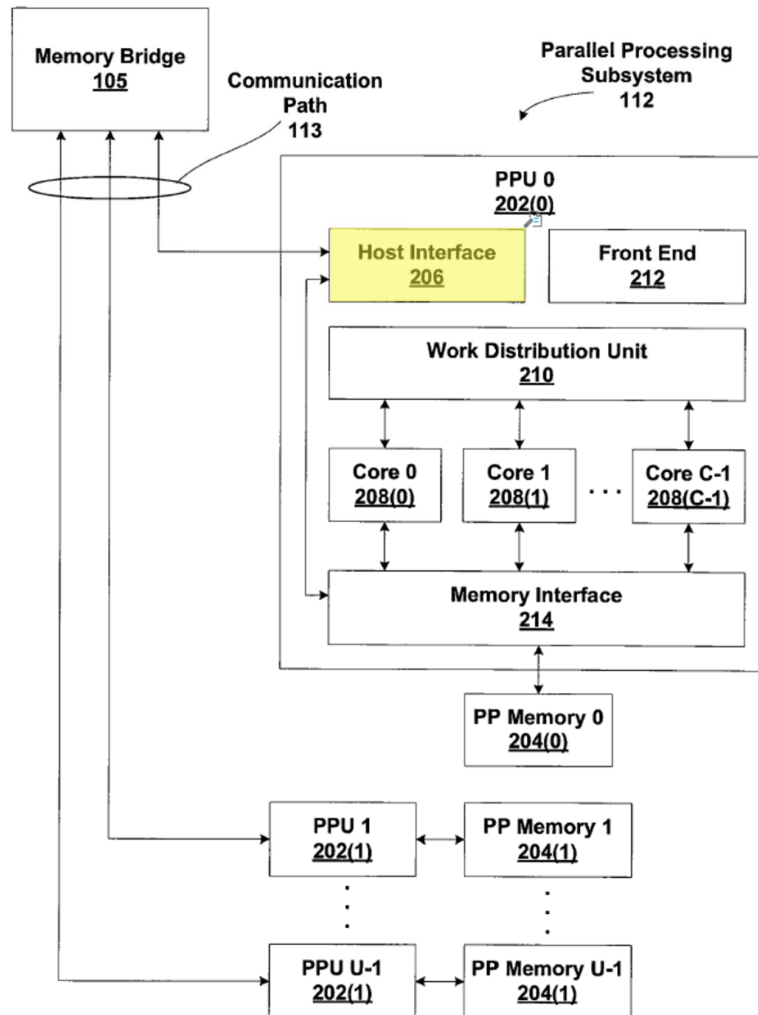


Figure 2

Ex. 1004 at Fig. 2 (annotated). Core interface 303 is not shown in Figure 2, but it is shown in Figure 3 as downstream of work distribution unit 210, which is shown to be downstream of host interface 206 in Figure 2 above. According to Dr. Pedram’s logic above regarding component can receive, “the work distribution unit 210, the front-end unit 212, and the core interface unit 303” (the alleged engines) in Wilt do

not receive the alleged tasks “from the scheduling module,” because that function of receiving has already been performed by the “host interface 206.”

59. Dr. Pedram testified that a POSITA would understand that the “host interface” was an engine, just like engines 202 in Figure 2 of the ’381 patent above, although he chose not to allege it as one. Ex. 2005 (Pedram 1/4/23 Dep. Tr.) at 275:10-276:10, 295:3-296:21, 298:21-299:10. In my opinion, Dr. Pedram declined to identify the host engine as a claimed engine because it would undercut his identification of downstream elements as the claimed engines.

c. The Alleged Engines Do Not Each Load State Data

60. Dr. Pedram only alleges that one of the three identified engines, core interface 303 loads state data: “Wilt discloses that core interface 303 loads the state parameters associated with each command thread into core 208 by loading parameters into parameter memory 308, which is part of core 208 of PPU 202.” Ex. 1003, ¶ 74. The other identified engines, namely the work distribution unit 210 and the front-end unit 212, are not alleged to, and in fact do not, load any state data.

2. Wilt Fails To Disclose or Render Obvious the Claimed “Shader Core”

a. The Alleged Shader Core Does Not Receive Tasks From The Alleged Engines

61. Claim 15 requires “a shader core configured to receive the plurality of tasks from at least one of the plurality of engines.” Ex. 1001 at Claim 15. As

explained above in Section IX.A.1.a, under both theories of what constitute the plurality of engines and the shader core, the alleged shader cores do not receive tasks from any of the engines, because the engines are within the shader core itself. This limitation is thus not taught for at least that reason.

b. The Alleged Shader Core Does Not Execute a First Task While Executing a Second Task

62. Wilt discloses only a single instruction, multiple-thread (“SIMT”) system (like the “SIMD” in Stuttard), which can only operate on a single instruction at a time. Stuttard states: “In a way similar to a SIMD machine, a SIMT parallel processing core 208 executes instances of a single parallel program on different data across a plurality of parallel processing engines 302 included in the core 208.” Ex. 1004 at 6:13-16. Meanwhile, Wilt states “[t]hus, for example, the core 208 may be configured to execute a series of common instructions on the parallel processing engines 302 within the core 208. The series of instructions to a single parallel processing engine 302 constitutes a thread, as defined previously.” Ex. 1004 at 6:16-21. Wilt continues to state that “each core 208 includes an array of P (e.g., 8, 16, etc.) parallel processing engines 302 configured to receive SIMT instructions from a *single* instruction unit 312.” Ex. 1004 at 6:36-39. At best, Wilt discloses multiple *threads* executing simultaneously. Thus, Wilt does not disclose that different *tasks* can be executed at the same or overlapping times. There is thus no disclosure that

different *tasks* can be executed at the same or overlapping times, let alone a configuration in the shader core of Wilt that could support simultaneous execution of different tasks. As explained above in Section VI.B, absent additional disclosure explaining which tasks certain threads correspond to, it cannot be assumed that any two or more threads will correspond to two different tasks.

63. Dr. Pedram confirmed that merely processing multiple *threads* simultaneously says nothing about whether the system can execute multiple *tasks* simultaneously. He testified “if a system can process multiple threads at the same time, it does not necessarily care about whether those threads belong to the same higher-level task or multiple different higher-level tasks.” Ex. 2005 (Pedram 1/4/23 Dep. Tr.) at 305:18-306:9. There is no disclosure in Wilt that different threads correspond to different tasks, much less *constitute* in some combination the different tasks that must be executed per this claim limitation, and Petitioner’s expert clarified there is no inherent reason why they would correspond to different tasks.

64. Even if Wilt could, by happenstance, execute two tasks simultaneously, that would be insufficient. There must be a specific design that causes the two tasks to be executed simultaneously (as Petitioner argued previously, see Section V), not that there is a “probability” that it will happen. There is no such disclosure of such an affirmative configuration in Wilt, nor has Dr. Pedram pointed to anything.

B. Ground 2: Jiao and the Knowledge of A POSITA Do Not Render Claims 15-20 of the '381 Patent Obvious

65. In my opinion, claims 15-20 are not obvious in view of Jiao and the knowledge of a POSITA.

1. Jiao Fails To Disclose or Render Obvious the Claimed “Plurality Of Engines”

a. The Alleged Engines Do Not Receive Tasks

66. Claim 15 of the '381 patent requires “a plurality of engines configured to receive . . . a plurality of tasks.” Ex. 1001 at Claim 15. But Petitioner’s alleged engines do not receive tasks. Dr. Pedram opines that a “POSITA would have understood the vertex shader 304, geometry shader 306, and pixel shader 310 in graphics pipeline 208 of GPU 114 to be “engines” as recited in claim 15. I disagree for the following reasons.

67. First, Jiao states that the alleged engines are “shader programs”: *A shader generally refers to a program* used in 3-dimensional (3D) computer graphics that may be used to determine the final surface properties of an object or image. Ex. 1005 at 1:47-50; *id.* at 1:43-47 (“These further transformations from eye space may include the use of programmable *shader programs*, such as a *vertex shader, pixel shader, and/or geometry shader.*”), 5:47-51. But the '381 patent teaches that in its terminology, “shader programs” are parts of “tasks”—they are not engines that *receive* tasks. More specifically, shader programs define the sequence of

instructions of threads, which themselves are parts of tasks, as explained above in Section VI.B. The '381 patent states that “[t]o 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, such as a geometry shader, a pixel shader, a vertex shader, or the like.”). Ex. 1001 at 1:47-52. The '381 patent further states, “[t]o efficiently execute the *various threads (e.g., shader programs)* of a single graphics-processing task, the GPU includes an array of processing elements, called a shader core.” *Id.* at 1:58-61. Thus, it is clear that the shader programs, including the exact ones identified by Dr. Pedram (vertex, geometry, and pixel), are constituent parts of tasks; they do not receive tasks.

68. Jiao is in agreement with the '381 patent. The parts of Jiao that Dr. Pedram ignores confirm, just as in the '381 patent and in the third party literature, *see* Section VI.D, that Jiao’s shader programs are programmed or configured ahead of time to perform specific actions on the data that they will receive while the system is running, not to receive and interpret “tasks.” Jiao refers to the shader programs as “**programmable** vertex 304, geometry 306, and pixel shaders 310.” Ex. 1005 at 9:25-27. Jiao further states that “geometry shader 306 . . . may be further **configured to perform various algorithms**, such as tessellation, shadow volume generation, etc.” Ex. 1005 at 6:33-35.

69. Jiao’s vertex shader 304, graphics shader 306, and pixel shader 310 are not block components of the GPU. To the contrary, Jiao confirms that vertex shader 304, graphics shader 306, and pixel shader 310 are “shader programs” that execute *on* the graphics processor, and not as “block components” *of* the graphics processor:

At the center of the programmable processing environment is the computational core 204, which processes various instructions and includes the CB system 200. 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.*

Ex. 1005 at 6:57-59.

70. Second, Dr. Pedram opines that “[t]hese shaders receive and process *graphics data (e.g., graphics primitives)* to produce pixel data.” Ex. 1003, ¶ 97. Graphics primitives are basic shapes. I agree that the only thing that Jiao discloses as being “received” by the shader programs is graphics data, including graphics primitives. Those plainly are not tasks, even under Dr. Pedram’s broad construction. That construction requires, at a minimum, some sort of instruction, command, etc.—a unit of work to be done or an application portion to be “executed.” But the only evidence is that the shader programs receive data, operate on it in a pre-programmed way, and move it along a serial graphics pipeline.

71. Jiao itself confirms the only things the shader programs receives are data. Figure 3 of Jiao shows the shader program operation in detail:

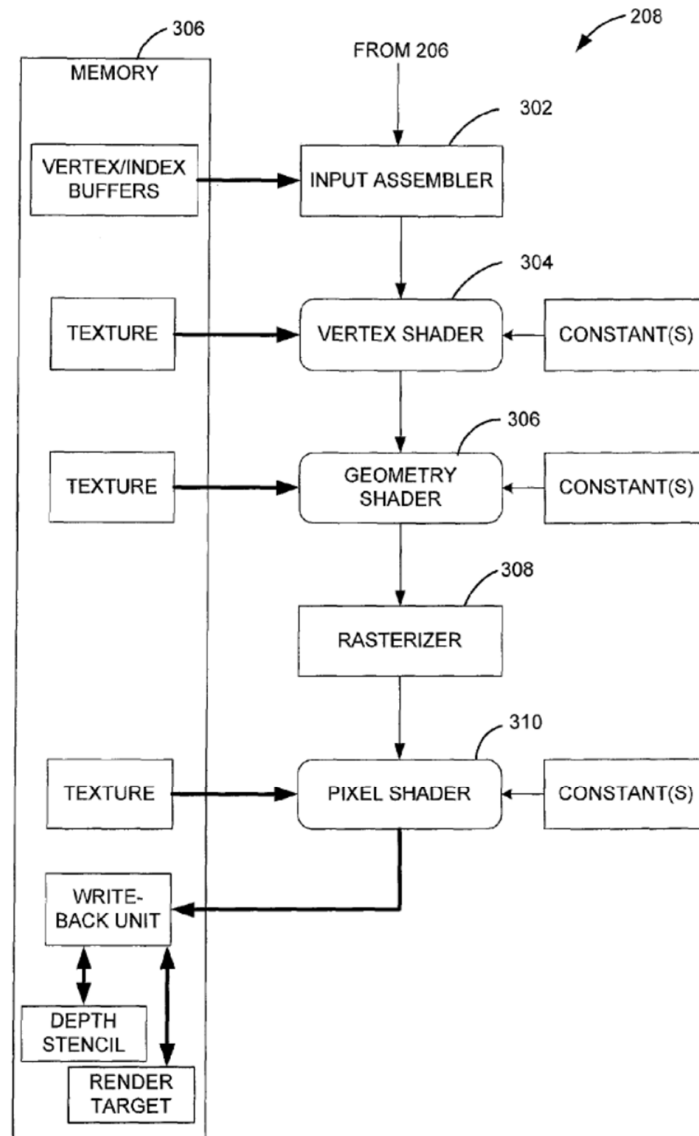


FIG. 3

Id. at Fig. 3. As shown above, the only inputs to the shader program chain 304, 306, and 310 are “textures,” “constants,” (neither of which are “tasks”) and what is sent from the “input assembler.” The only mention of “input assembler” in the specification states that it “essentially fetches (or generally receives) all vertex data

from the memory 312 using index buffers, and assembles primitives [*i.e.*, basic shapes] for further processing in the computational core 204.” *Id.* at 6:13-17. None of those items are tasks, and Jiao’s specification confirms on a one-by-one basis that each of the shader programs merely receives data or primitives:

- “The vertex shader 304, as is known, processes vertices.” *Id.* at 6:22.
“[V]ertex information is passed to vertex shader 304 in the computational core 204 to implement vertex transformations.” *Id.* at 7:56-60.
- “The geometry shader 306 receives, as inputs, vertices for a full primitive. . . The geometry shader 306 also receives constants from the constant buffer.” *Id.* at 6:29-34.
- “The pixel shader 310 also receives constants from the constant buffer of the constant buffer system 200, and operates to perform interpolations and other operations that collectively determine pixel colors for output to a write-back unit in memory 312.” *Id.* at 6:40-44.
- “The Z-values are passed back to the pixel shader 310.” *Id.* at 8:7-10.

See generally id. at 6:1-48.

72. Jiao has no disclosure of a shader program receiving a “task,” or anything other than graphics data and primitives. The commands and instructions that might make up a thread, and at a higher level a task, are interpreted by another component in the system be-sides the shader programs (although it is not clearly

stated what) and then only data is input to the vertex shader, which is the start of the shader program chain in Figure 3, and then to subsequent shader programs: “***Responsive to a command*** from the driver software 128, such as to draw a triangle, ***vertex information is passed to vertex shader 304*** in the computational core 204 to implement vertex transformations.” Ex. 1005 at 7:57-60. In light of the above disclosure, it is clear that the shader programs do not receive any “unit of processing work or portion of an application to be executed,” the construction for “task”; they just receive data.

73. All of the above is true for both of Dr. Pedram’s alleged scheduling modules. In one theory, Dr. Pedram opines the shader programs receive tasks both from the EU Pool Control 206 (Ex. 1003, ¶ 105), as the alleged scheduling module, and in another theory through a chain of components (including EU Pool Control 206) from the “driver software 128” (Ex. 1003, ¶ 106), as the alleged scheduling module. But neither theory contradicts the above structure in Jiao regarding what ultimately gets received by the shader programs.

74. Third, to the extent Dr. Pedram opines that Jiao’s shader programs, output, and therefore must receive, “tasks,”¹ Dr. Pedram omits that Jiao discloses

¹ As Dr. Pedram acknowledges, Jiao uses the term “tasks” inconsistently from the ’381 patent to refer to “threads” that are output from its shader program. Ex. 1003, ¶ 103 (“At times, Jiao could be read to conflate the terms, i.e. uses the term ‘tasks’ and ‘threads’ interchangeably.”).

that the shader programs actually “generate” those tasks. Ex. 1005 at 9:35-37. It is not possible for the shader programs to “generate” the tasks if they already existed and were supplied to the shader programs by Dr. Pedram’s alleged scheduling module. Dr. Pedram’s inference that since “tasks” are output by the shader programs, they must also be received by the shader programs fails because the shader programs themselves are the constituent parts of a single task. There would be no reason for the shader program to receive any tasks or instructions all of the tasks and instructions that define a shader program are pre-programmed and compiled into the machine code of a shader program at the time of its creation.

Dr. Pedram’s Alternative Theory Including “Thread Slots”

75. Dr. Pedram’s alternative argument adds “the ‘thread slots’ within the execution units 420 of the EU pool 414” to the plurality of engines alleged in its main embodiment. Ex. 1003, ¶ 98 (“In the alternative, the claimed ‘engines’ are the vertex shader 304, geometry shader 306, and pixel shader 310 in graphics pipeline 208 of GPU 114 *plus the “thread slots” within the execution units 420 of the EU pool 414.*”). Dr. Pedram opines that the shader programs generate and provide to the thread slots “shader tasks.”

76. This argument, even if true, would still fail to disclose the claimed element for multiple reasons: (1) each engine must receive tasks according to the claim language, as explained below, and the shader programs do not receive tasks,

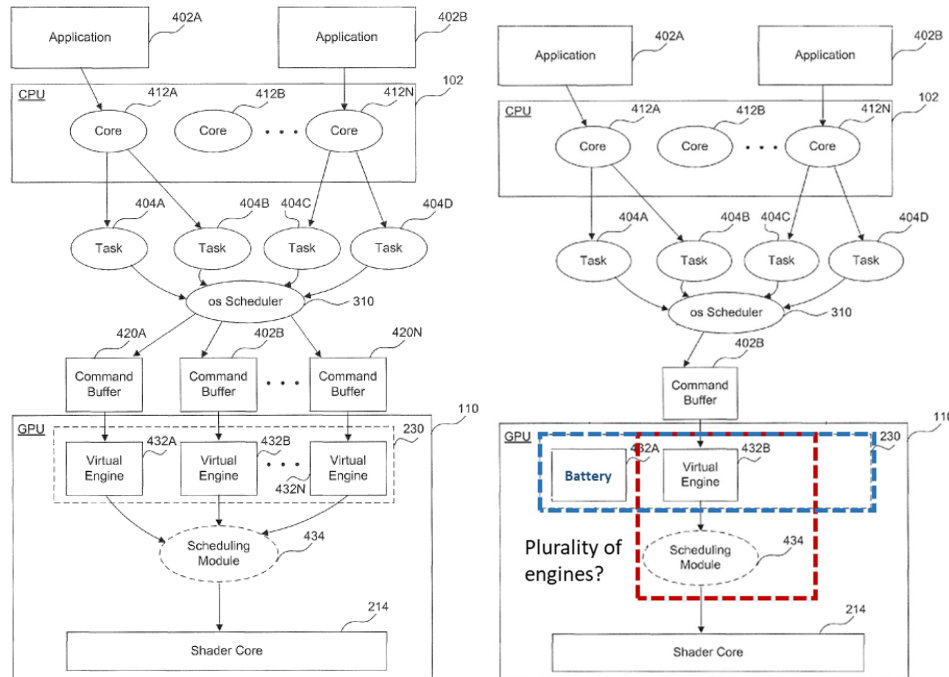
as explained above, and (2) the thread slots would need to receive the “shader tasks” from a scheduling module according to the claim language, but the thread slots receive the so-called shader tasks exclusively from the shader programs, which generate the shader tasks in the first instance—and the shader programs are not alleged to be scheduling modules. Ex. 1005 at 9:35-37.

77. As to point (1), claim 15 requires “a plurality of *engines* configured to receive, from a scheduling module associated with a second processing unit, a plurality of *tasks*.” The plain and ordinary meaning of that phrase means that each engine must receive tasks from a scheduling engine; it is not sufficient for just one engine or a subset to receive tasks. Further, the entirety of the specification of the ’381 patent explains that each of the engines receives tasks. *See, e.g.*, Figs. 2-4, 5:4-13, 6:65-7:23, 7:25-62. The claim language simply states that multiple of the engines are required.

78. To explain further, claim 15, by its own terms, requires the plurality of engines, themselves, to be configured to independently receive a plurality of tasks from a scheduling module, and to load associated state data for those tasks. Ex. 1001 at Claim 15 (“15. A computing system comprising...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[.]”). This means that each of the claimed engines must independently

receive at least one task from a scheduling module, and load state data associated with such task. A system in which just one of the alleged plurality of engines is configured to receive tasks from a scheduling module and to load associated state data cannot satisfy the claim's express requirement that the plurality of engines, themselves, must be configured to receive a plurality of tasks from the scheduling module and to load associated state data. In my opinion, a construction where only one of the plurality of engines needs to perform the claim's requirements reads the configuration of (*i.e.*, a design or implementation that achieves the stated effect) "the plurality of engines" out of the claim.

79. The '381 patent's specification further compels the above construction. The '381 patent's specification teaches that although the plurality of engines, identified as 432A, 432B, etc. in Fig. 4 (reproduced below), are grouped together within a "command processor" 230, each of the plurality of engines are configured to independently receive a plurality of tasks from a scheduling module (identified as OS scheduler 310), and to load associated state data. Ex. 1001 at Fig. 4. Specifically, each of the plurality of engines services a respective command buffer to independently receive at least one task from the OS scheduler 310. As a result, each of the plurality of engines are configured to independently receive a plurality of tasks from the scheduling module and to load associated state data.



Ex. 1001 at Fig. 4 (annotations shown in diagram on the right).

80. The annotated version of Figure 4, juxtaposed on the right hand side of original Figure 4 above, shows the effect of only requiring one of the engines to perform the claimed limitation. In a first hypothetical (A - blue), the command processor 230 (boxed in blue), has been annotated to include only a single engine 432B, which serially receives a plurality of tasks from a scheduling module (OS Scheduler) and loads associated state data, and a battery 432A that provides power to the engine 432B. Under the construction where only one engine is required to perform the claimed limitation, the annotated command processor 230 (comprising the battery 432A and engine 432B) would qualify as the claimed “plurality of engines configured to receive a plurality of tasks and to load associated state data”

because the battery 432A and engine 432B, evaluated as a group, receive a plurality of tasks and load associated state data. This is even though, beyond supplying power to the actual engine 432B, the battery 432A has nothing to do with receiving tasks or loading state data.

81. In a second hypothetical (B - red), the alleged plurality of engines are boxed in red and identified as engine 432B and scheduling module 434. As in the prior hypothetical, single engine 432B, which serially receives a plurality of tasks from a scheduling module (OS Scheduler) and loads associated state data. The tasks are then serially passed to scheduling module 434, which schedules the tasks for serial execution on the shader core 214. Even though scheduling module 434 has nothing to do with loading associated state data (which is handled by the single engine 432B), under the construction where only one engine is required to perform the claimed limitation, the scheduling module 434 can nonetheless be one of the engines that make up the plurality of engines because when evaluated as a group, single engine 432B and scheduling module 434 receive a plurality of tasks and load associated state data.

82. There is no reasonable interpretation of the claims and specification that would lead a POSITA to believe that either of these hypotheticals (A) or (B) are what the inventors intended as the claimed “plurality of engines.” To the contrary, in describing the configuration of the plurality of engines to receive a plurality of

tasks from a scheduling module, and to load associated state data, the '381 patent explicitly contrasts the claimed engines' ability to independently receive a plurality of tasks over "conventional" graphics processing systems that received and processed tasks serially. The passage below from the '381 patent specification, which describes the configuration of the claimed plurality of engines, and with reference to Figure 4, is illustrative:

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 applications 402. For example, OS scheduler 310 provides a first type of task (e.g., high-priority tasks) to first command buffer 420A, a second type of task (e.g., graphics-processing tasks) to second command buffer 420B, and so on.

GPU 110 includes a plurality of virtual engines 432A-N, each configured to service one of command buffers 420A-N. For example, a first virtual engine 432A is configured to service first command buffer 420A; a second virtual engine 432B is configured to service second command buffer 420B; and an N-th virtual engine 432N is configured to service N-th command buffer 420N. Tasks from virtual engines 432 are then (substantially) simultaneously executed by shader core 214 as described above.

Id. at 13:10-22:41; see also *id.* at 6:65-7:19, 9:34-369:47-49, 10:25-39, Figs. 2, 3A, 3B, and 4.

83. The '381 patent specification's disclosure above is but one example of multiple statements by the inventors distinguishing over "conventional" systems in

which the tasks were received serially, through a single command buffer, and therefore could not be executed in parallel on a single shader core. *See id.* at 2:8-13. (“To 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.”), 4:48-55 (“Each virtual engine is configured to receive data-parallel processing tasks (e.g., graphics-processing tasks and general-compute tasks) and independently execute these tasks on the single shader core. In this way, the processing unit may execute two or more different streams of processing tasks—such as, a first stream of low-latency processing tasks and a second stream of standard graphics-processing tasks—without requiring a context switch.”).

84. Reading out of the required configuration of each engine would allow for any conventional pre-processing blocks to be grouped with a single engine to satisfy the claim. For example, Figure 2 of the '381 patent (reproduced and annotated below) identifies conventional downstream pre-processing elements, such as a Vertex Analyzer 208 and Scan Converter 212, and differentiates their functionality from those of the engines.

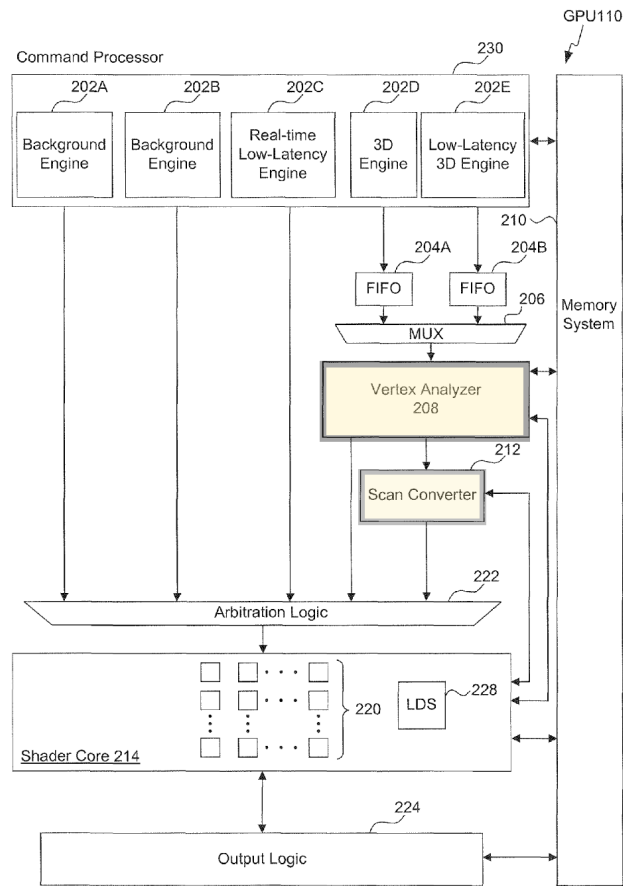


FIG. 2

Id. at Fig. 2 (annotated), 7:34-62. The '381 patent teaches that “Vertex Analyzer 208 identifies shader programs associated with a graphics-processing and/or general-compute task and schedules when each shader program can be launched in shader core 214 based on input and output data that will be available,” and that “Vertex Analyzer 208 sends graphics-processing tasks to scan converter 212 and sends general-compute tasks to arbitration logic 222.” *Id.* at 7:42-46, 7:55-57.

85. But as acknowledged by Dr. Pedram, during his deposition, pre-processing blocks such as the Vertex Analyzer 208 and Scan Converter 212 in Figure 2 cannot be an “engine.” Ex. 2005 (Pedram 1/4/23 Dep. Tr.) at 84:9-88:12. Dr. Pedram stated that would be “a ridiculous position” because “then everything that you see in [the] Figure 2, every component here you see . . . would be an engine.” *Id.* at 89:21-91:25.

Q What -- what is the attribute of an engine that is -- that is missing from vertex analyzer [of Figure 2] that disqualifies it from -- from being an engine? MR. JOHNSON: Objection; form.

THE WITNESS: At least two. One, it's not virtual; two, it's not receiving it from the scheduling module. The function has already been performed. You're trying to associate a duplicative function to a component in the system, and no person of -- skilled in the art, no one that understands the cost associated with building chips or -- or -- or providing functionality and services to systems that make use of this chip would understand that this duplicative functionality would be implemented in practice. It's just nonsensical to do it. If I already have the function, why would I want to repeat that same function again? I wouldn't do it. It's done. The engines already have done what they needed to do. I have the -- the data, the task received; I've loaded the state data that I need to it, I've created this package. I'm sending it to the vertex analyzer, which is a hardware component, whereas these engines are virtual, software, as far as my understanding would be, although I -- I know the claim construction from the Court, so -- so, yeah, it doesn't work like that.

Id. at 87:9-88:12.

MR. RIZK: So -- so I'm not interested in your opinions, at this point, about why you believe that the engines need to be software, need to be virtual. So the only reason that you've provided that is within the -- the

way the Court has constructed engines, that preprocessing logic isn't an engine, is that it doesn't receive the tasks from a scheduling logic.

Q Do I understand correctly?

A Not only that. This -- this duplicative functionality argument is an important one, in my view, because whenever you apply a principle and then at the end it becomes ridiculous, you should give up on that principle, right? So if I take the view that any component, whether it's hardware or software, that receives a task can be included either directly or indirectly, as you have said, as long as I'm coupling to it somehow indirectly, it's receiving the task -- right? -- according to what you have posed to me, then everything that you see in Figure 2 embodiment, every component here you see -- right? -- perhaps with the exception of the memory system, would be an engine. Again, that's a ridiculous position to take for anyone. No POSITA would assume that's the meaning of an engine. You could say it; but, again, it's not a matter of what you say, it's a matter of what a POSITA would understand at the time of the invention; and no one would assume it because there is a separate element related to the shader core. According to your definition, shader core would be also an engine. Why not?

Id. at 90:17-91:25.

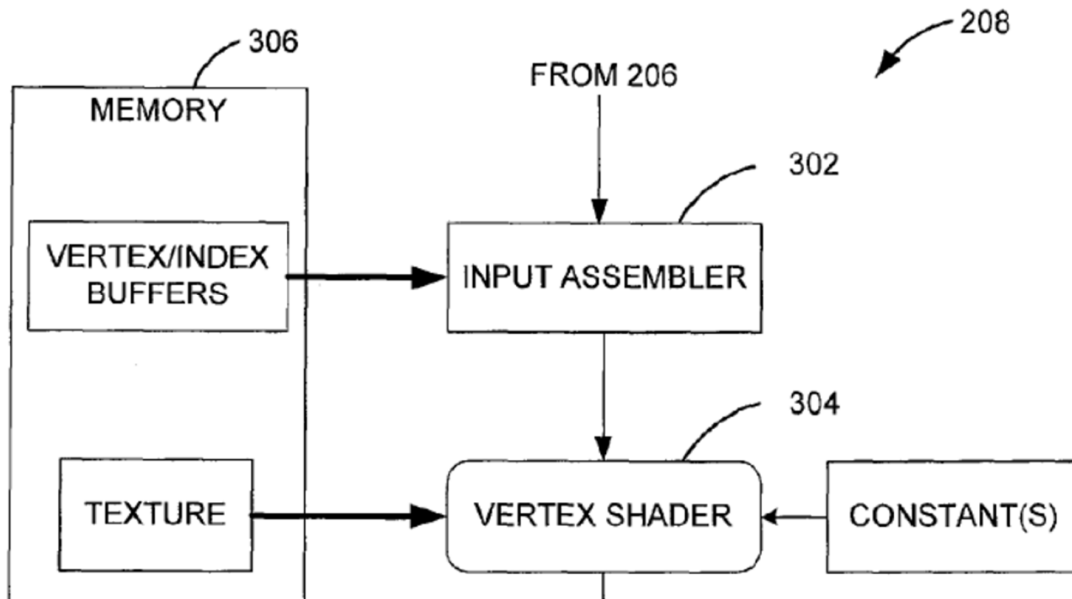
86. Similarly, regarding Stuttard, Dr. Pedram testified that, because the thread manager 102 allegedly performs all of the functionality of the plurality of engines to receive a plurality of tasks and to load associated state data, per his earlier discussion on Figure 2, it would be "ridiculous" for a POSITA to consider any downstream components, such as array controller 104 or channel controller 108 as also constituting one of the plurality of engines. *See id.*; *see also id.* at 272:15-24, 273:20-23. Instead, according to Dr. Pedram, the array controller loads the state data for each task. *Id.*

87. Yet, this approach which Dr. Pedram characterized as “ridiculous”—and which is also depicted in hypotheticals (A) and (B) of a conventional system above—equates to construing the plurality of engines limitation in Claim 15 as being satisfied even if only one of a “plurality of engines” is configured to receive tasks and load state data. This construction has the same problem that the inventors and the patent intended to solve in conventional systems. Thus, a construction of the plurality of engines as requiring anything less than all of the plurality of engines to be configured to receive at least one task from a scheduling module and to load state data associated with that task, is divorced from the specification’s clear teachings of their advance over conventional systems.

b. The Alleged Engines Do Not Receive Tasks From the Alleged Scheduling Module

88. A POSITA would understand that claim 15 requires that each engine receive a plurality of tasks from a scheduling module. Dr. Pedram admitted that when there is a series of elements in a processing chain, only the first one “receives” the alleged processing task from the alleged scheduling module. *See* Section IX.A.1.b. Downstream elements do not receive alleged tasks from the alleged scheduling module, as that action has already been completed.

89. But Jiao provides, as shown in Figure 3, the alleged engines are in a chain, where the vertex shader feeds its output to the geometry shader's input, which in turn feeds its output to the pixel shader input. Ex. 1005 at Fig. 3.



Id. Fig. 3 (excerpted). According Dr. Pedram, the input assembler would be the sole item to receive the alleged “tasks.” The vertex shader and the other downstream elements, the geometry shader and pixel shader, may receive the alleged tasks, but they do not do so from the alleged scheduling modules, which are EU Pool Control 206 and “driver software 128.”

90. Moreover, as explained above in Section IX.B.1.a regarding Dr. Pedram’s alternative identification of “thread slots” as engines on pages 49-59, claim 15 requires that *each* engine receive tasks from a scheduling module. Even if the first alleged engine in the chain, vertex shader 304 received the alleged tasks from

the alleged scheduling module, the downstream components, the geometry and pixel shaders, and under Dr. Pedram's alternative theory, "thread slots" in the execution units 420, do not receive alleged tasks from the alleged scheduling modules—because one alleged engine, the vertex shader, has already completed that task, according to Dr. Pedram.

91. Separately, as to Dr. Pedram's opinion that "Driver software 128 (running on CPU 126)" is a "scheduling module," Ex. 1003, ¶ 106, the alleged engines, the shader programs, are not connected to driver software 128 and do not receive anything from them, let alone "tasks." What Jiao actually states is that "[r]esponsive to a command from the driver software 128, such as to draw a triangle, vertex information is passed to vertex shader 304 in the computational core 204 to implement vertex transformations." Ex. 1005 at 7:565-8:14. This portion of Jiao states nothing more than what is explained above: the alleged engine, the vertex shader 304, receives "vertex information." Vertex information is not even a thread, let alone a full task; it is instead vertex data to be operated on by the vertex shader.

c. The Alleged Engines Are Not Separate From the Alleged Shader Core

92. As explained above in Section IX.A.1.a, claim 15 requires that the plurality of engines and the shader core be separate. Dr. Pedram has two theories as

to what satisfies the “plurality of engines” requirement, but both point exclusively to things within the alleged shader core, and so both fail.

93. Dr. Pedram’s first theory is that a “POSITA would have understood the vertex shader 304, geometry shader 306, and pixel shader 310 in graphics pipeline 208 of GPU 114 to be ‘engines’ as recited in claim 15.” Pet. at 39. The alleged shader core is EU Pool 414. Ex. 1003, ¶ 110 (“A person of ordinary skill in the art would have understood Jiao’s EU pool (EUP) 414 to be a ‘shader core’ as recited in claim 15.”). But the shader programs are within EUP 414, in contradiction to the claim language requiring they be separate. Although Figure 4 displays shader programs outside of EUP 414, the specification clarifies that the shader programs are actually located inside EUP 414, despite the figure: “In general, an embodiment of the graphics processor 202 utilizes programmable vertex 304, geometry 306, and pixel shaders 310. *Rather than* implementing the functions or operations of these components as *separate fixed function units* with different designs and instruction sets, *the operations are instead executed by the pool 414* of execution units 420a, 420b . . . 420m with a unified instruction set.”). Ex. 1005 at 9:25-31.

94. This is exactly how the ’381 patent describes shader programs as well: they execute on the shader core—they are not separate from it: “processing elements 220 [in Shader Core 214, Fig. 2] may implement a plurality of different shader programs (e.g., geometry shader, vertex shader, pixel shader, tessellation shader, or

the like).” Ex. 1001 at 8:34-38. Jiao and the ’381 patent are thus in agreement that shader programs such as ones alleged by Petitioner to be “engines” are run within the alleged shader core and are not separate from it as required.

95. Dr. Pedram’s second alleged “plurality of engines” is “the vertex shader 304, geometry shader 306, and pixel shader 310 in graphics pipeline 208 of GPU 114 *plus the ‘thread slots’ within the execution units 420 of the EU pool 414.*” Ex. 1003, ¶ 98. This merely adds a component that is within the EUP 414 and so this group of alleged engines fails for the same reasons as above.

2. Jiao Fails To Disclose or Render Obvious the Claimed “Scheduling Module” and “Second Processing Unit” Because the Alleged Scheduling Modules Are Not Associated With the Alleged Second Processing Units

96. Claim 15 of the ’381 patent requires “a scheduling module associated with a second processing unit,” as the ID found, “prosecution history disclaimer occurred which limited the scope of ‘associated with,’” whereby mere connection to an alleged second processing unit and receipt of data from it does not suffice. Ex. 2003 (ID) at 151-55. Rather, the scheduling module needs to be “within” the second processing unit. *Id.* During prosecution of the ’381 patent, the applicant avoided prior art (a different Jiao reference) by arguing that the alleged scheduling module, “Thread Scheduler 210” was within the alleged “first processing unit,” not the required “second processing unit,” and therefore was not “associated with” any

second processing unit— despite it being externally connected to other items such as “Graphics Programs” 1-N and receiving threads from it:

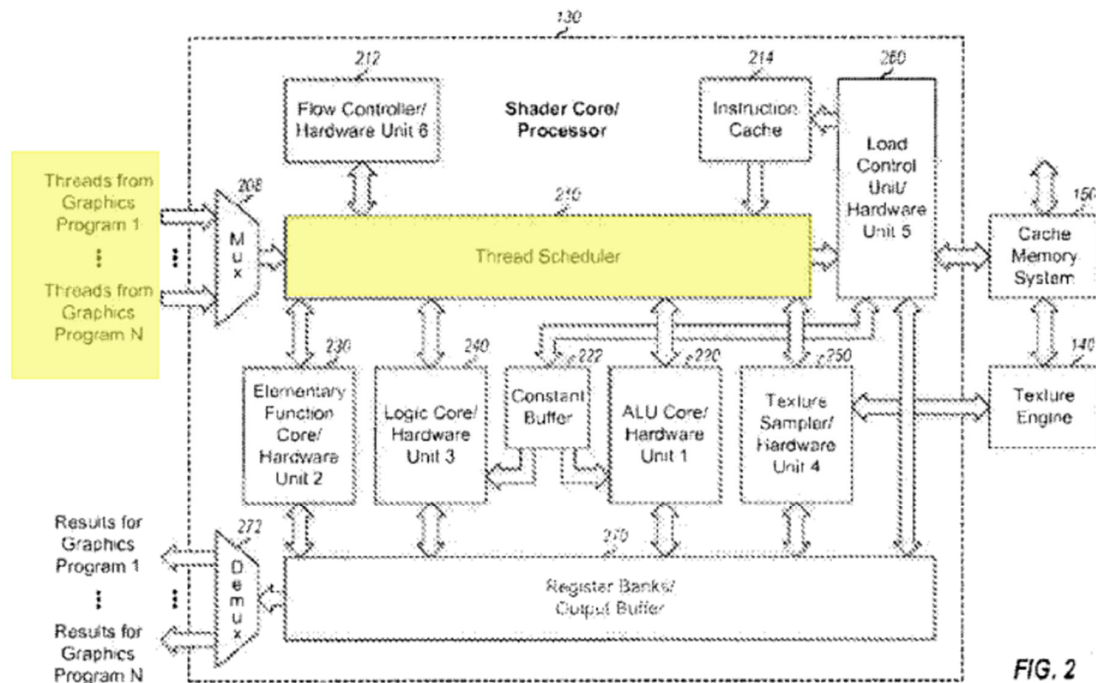


FIG. 2 of Jiao

Ex. 1002 (‘381 patent prosecution history, March 13, 2014 remarks) at 183 (annotated). As the ID held, “[t]he only substantive reason given for why the applied reference did not teach the limitation is that the reference’s thread scheduler 210 is ‘*within*’ the shader core, *and therefore ‘associated with’* that shader core, when it needed to be associated with a different ‘second processing unit.’” Ex. 2003 (ID) at 155. The ID thus held there was a disclaimer of scope to avoid mere connection and to require being “within” the alleged second processing unit. *Id.*

97. Dr. Pedram opines that Pool Control Unit 206 is a “scheduling module,” and “is associated with the command stream processor 406, which is the claimed ‘second processing unit,’” Ex. 1003, ¶ 105, because it is connected to command stream processor 406 and receives data from it:

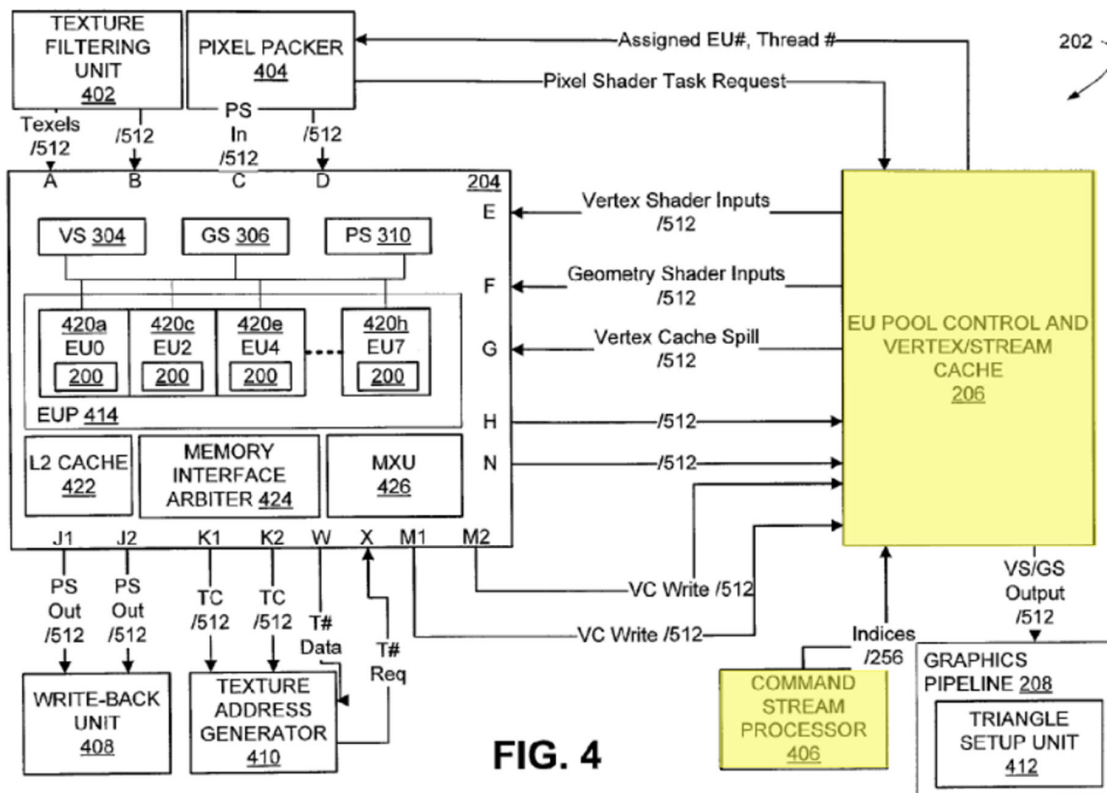


FIG. 4

Ex. 1005 at Fig. 4 (annotated). But this mere connection fails to meet the “associated with” requirement of the claims, as explained above.

98. Dr. Pedram identifies as the alleged second processing unit what the applicant specifically avoided in Figure 2 of the prior Jiao reference used during

prosecution—a separate but connected entity (“Graphics Program”) from which the alleged scheduling engine (“Thread Scheduler 210”) receives data. If mere connection to a separate component and reception of data from it would have sufficed, the “Graphics Program” in the Jiao reference used in prosecution would have been a second processing unit and the “Thread Scheduler 210” would have been a scheduling module, but neither was true. Instead of receiving data from the second processing unit, the scheduling module needs to be within it, but Dr. Pedram has not made this showing.

99. Similarly, Dr. Pedram opines that driver software 128 is the claimed scheduling module, and it is associated with CPU 126. Ex. 1003, ¶ 106. But driver software is within a different component from CPU 126; it is in system memory 124, which is separate from CPU 126. Ex. 1005 at 5:16-23 (“The system memory 124 also comprises driver software 128”); Fig. 1. Therefore, driver software 128 fails as the alleged scheduling module as well.

3. Jiao Fails To Disclose or Render Obvious the Claimed “Shader Core”

a. The Alleged Shader Core Does Not Execute A First Task While Executing A Second Task

100. Dr. Pedram opines this claim limitation is met because Jiao states that “EU pool 414, at its peak, can process multiple *threads* simultaneously or substantially simultaneously.” However, as explained above in Section VI.B, and

as agreed by Dr. Pedram, a task is not a thread. Ex. 1003, ¶ 102. Further, as also explained above in Section VI.B and Section IX.A.2.b, as Dr. Pedram agreed, absent additional disclosure explaining which tasks certain threads correspond to, it cannot be assumed that any two or more threads will correspond to two different tasks. Therefore, processing multiple threads simultaneously is not sufficient to disclose the requirement to “execute a first task from the plurality of tasks while executing a second task.”

101. To the extent Jiao refers to a shader program generating “shader tasks,” and to the extent Dr. Pedram opines those tasks are executed by the alleged shader core, Jiao is using the term “task” differently from the ’381 patent. As explained above in Sections V and VI.B, the sub-portions of a “task” generated by the shader programs are only “threads,” and not a full “task.” Dr. Pedram previously testified that these low-level “shader tasks” generated by the shader programs, which are really threads, are not “tasks” as that term is used by the ’381 patent. Ex. 2005 (Pedram 1/4/23 Dep. Tr.) at 300:18-301:7 (“When the ’381 patent refers to “tasks,” it’s not referring to the types of lower-level tasks that might be generated by a shader program, correct? A Yes.”). Dr. Pedram agrees that the use of “task” in Jiao is inconsistent. Ex. 1003, ¶ 103 (“At times, Jiao could be read to conflate the terms, i.e., uses the term ‘tasks’ and ‘threads’ interchangeably.”). Using the terms how the ’381 patent does, these “shader tasks” are only threads. For example, the ’381 patent

itself discloses that “[t]o **perform a single** graphics-processing **task**, a GPU may execute **a plurality of different threads** (sequence of instructions).” Ex. 1001 at 1:47-51.

102. The mere configuration of a shader core to execute a first thread for an overlapping period of time while executing a second thread, however, does not, in and of itself, satisfy the claim language because the first and second thread may pertain to the same task. This is because a single task is ordinarily comprised of a plurality of threads. So the mere configuration of a shader core to execute a plurality of threads does not make it configured to execute a plurality of tasks. The ’381 patent specification describes this important point in detail.

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, such as a geometry shader, a pixel shader, a vertex shader, or the like. Each thread (e.g., shader program) is typically associated with a set of state data (such as texture handles, shader constants, transform matrices, or the like) that is locally stored in data-storage units of the GPU. The locally stored state data is called a context.

To efficiently execute the various threads (e.g., shader programs) of a single graphics-processing task, the GPU includes an array of processing elements, called a shader core.

Id. at 1:49-61.

103. The ’381 patent then goes on to describe how, in “conventional” GPUs, the threads of such tasks were delivered for execution by the shader core.

To 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.”

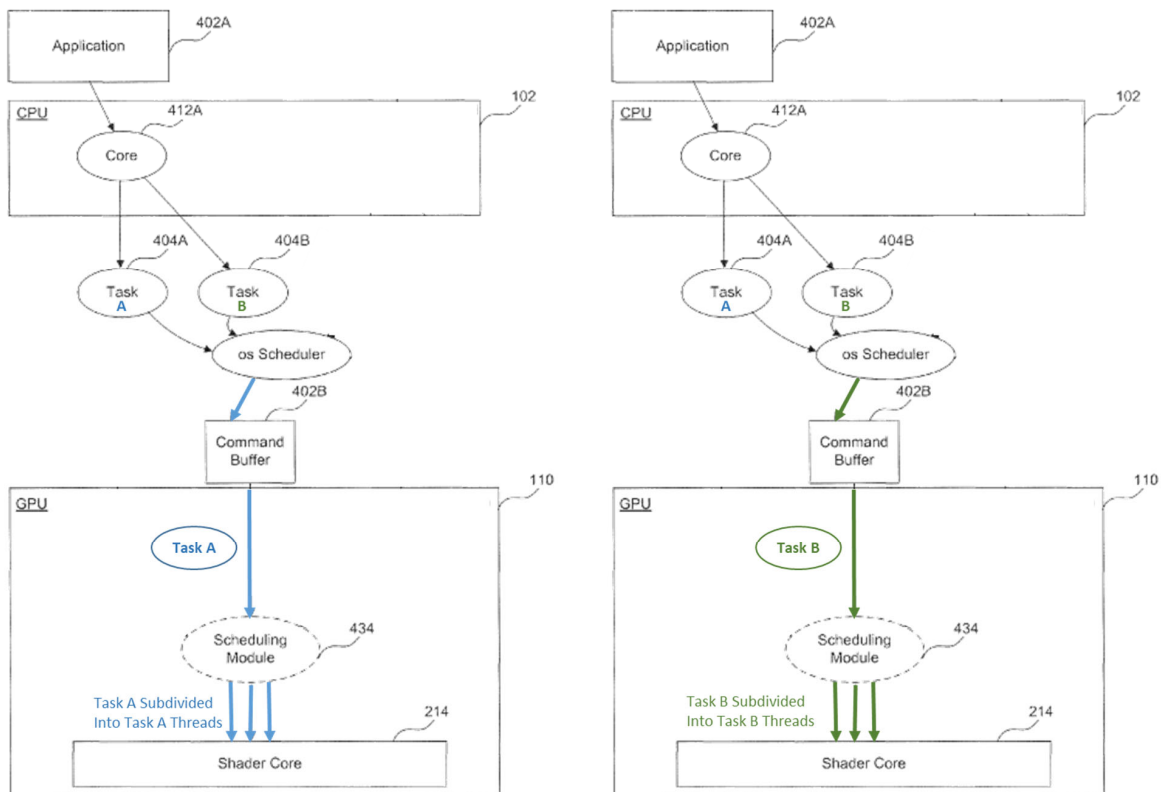
Id. at 2:8-14.

To perform the more-important (e.g., low-latency) task before the first task has finished completing in the shader core of the GPU, a conventional GPU performs a context switch. That is, the state data associated with the threads of the first task are swapped into back-up storage units maintained by the conventional GPU, and new state data associated with the threads (e.g., shader programs) of the more-important (e.g., low-latency) task are retrieved and placed in the data-storage units of the shader core. The shader core then executes the threads (e.g., shader programs) of the more-important (e.g., low-latency) task based on the new state data in the data-storage units.

Id. at 2:20-31.

104. To help further illustrate *a conventional system's* operation, Figure 4 of the '381 patent has been annotated (with some components removed) in the diagram below. As shown, an application 402A generates Task A (404A) and Task B (404B). *Id.* at 13:16. “OS scheduler would provide tasks 404 to a single command buffer [402B] in a serial fashion, and a conventional GPU [110] would serially process the tasks.” *Id.* at 10:23-25. “The shader core then executes the threads (e.g., shader programs) of [] [the] task[s].” *Id.* at 2:28-36. Thus, in a first phase, shown on the left, the OS Scheduler issues Task A to the GPU, which then processes Task A by subdividing Task A into a plurality of threads and executing them on the shader core. Only after this processing is complete, OS Scheduler supplies Task B to the

GPU, which processes the task in a similar fashion. Thus, even though the shader core 110 of the conventional system is configured to execute the plurality of threads within a single task (Task A or Task B) at the same time, the conventional shader core is incapable of executing a plurality of threads pertaining to different tasks at the same time (e.g., executing a first thread corresponding to Task A while, at the same time, executing a second thread corresponding to Task B). *Id.* at 10:22-25.



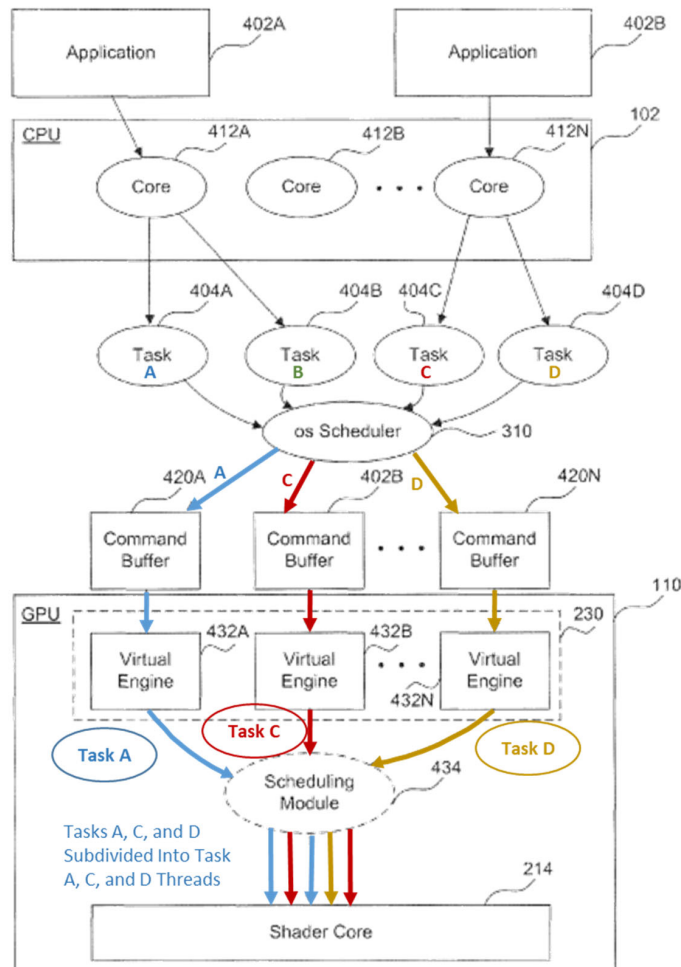
Id. at Figure 4 (annotated).

105. Unlike the conventional system operation as described above, the claim further requires “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.” *Id.* at Claim 15. The claim’s use of the definite article “the” to refer to the phrase “from the plurality of tasks” ties the first and second tasks to two separate tasks of the plurality of tasks received from the scheduling module. This means, in the context of executing tasks by executing their threads, that the shader core must be configured to execute a first thread that is part of a first task received from the scheduling module while executing a second thread that is part of a separate, second, task received from the scheduling module. *Id.*

106. To further illustrate ***the claimed system*** in operation, Figure 4 (in its entirety) has been annotated in the diagram below. As depicted, the plurality of engines 432A, 432B, and 432N, receive a plurality of tasks, identified as Task A (404A) in blue, Task C (404B) in red, and Task D (404N) in Orange, from a scheduling module (os Scheduler 310). *Id.* at 10:33-39. “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.” *Id.* at 10:22-28. But “[u]nlike [a] conventional system, OS scheduler 310 provides each task 404 to one of a plurality of command buffers 420A-N[.]. *Id.* The engines are “each configured to service one of command buffers 420A-N.” *Id.* at 10:33-39. Tasks [A,

C, and D] from [the] engines 432 are then (substantially) simultaneously executed by shader core 214 as described above.” *Id.* at 10:39-41.



Id. at Fig. 4 (annotated).

107. With regards to the threads of a given task, the '381 patent further discloses that “[e]ach virtual engine is configured to execute a stream of processing tasks provided by an OS scheduler, wherein each processing task of a given stream may include a plurality of individual processing threads... GPU 110

(substantially) simultaneously executes tasks from a plurality of streams, which correspond to a plurality of different contexts, in a single shader core by sharing resources of the shader core among the tasks.” *Id.* at 6:43-47. In other words, the ’381 patent discloses that the shader core substantially simultaneously executes a plurality of tasks by simultaneously executing threads of different tasks. *Id.* The configuration of the shader core is shown in the annotated version of Figure 4 above, with the scheduling module 434 providing threads of Task A (in blue), threads of Task C (in red), and a thread of Task D (in orange) to the shader core 214 for simultaneous execution.

108. Moreover, even if pointing to a thread were sufficient, Dr. Pedram has not shown that the system is “configured to” execute threads that were decomposed from two separate tasks. Any sort of “happenstance” or “probability” based argument, which is not supported in the first instance, would be insufficient under Dr. Pedram’s own view of the claims, as explained above in Section V.

b. The Alleged Shader Core Does Not Receive “The Plurality of Tasks”

109. Claim 15 requires that the shader core be configured to receive “*the* plurality of tasks from at least one of the plurality of engines,” where “the plurality of tasks” has antecedent basis in the previous element, “a plurality of engines configured to receive, from a scheduling module . . . a plurality of tasks.” Ex. 1001

at Claim 15. But in Jiao, the shader core does not receive anything received by the shader programs. They receive from the shader programs an alleged shader task that *the shader programs themselves generate*. Dr. Pedram cites to Jiao stating “[a]s various shader tasks are *generated by* the vertex shader 304, geometry shader 306, and pixel shader 310, they are delivered to the respective execution units 420 to be carried out.” Ex. 1003, ¶ 110; Ex. 1005 at 9:35-38. Thus, the execution units 420 in EUP 414 receive something that did not even exist until the shader programs generated them, which cannot be “*the plurality of tasks*,” the tasks that the alleged engines received.

C. Ground 3: Stuttard and the Knowledge of A POSITA Do Not Render Claims 15-20 of the '381 Patent Obvious

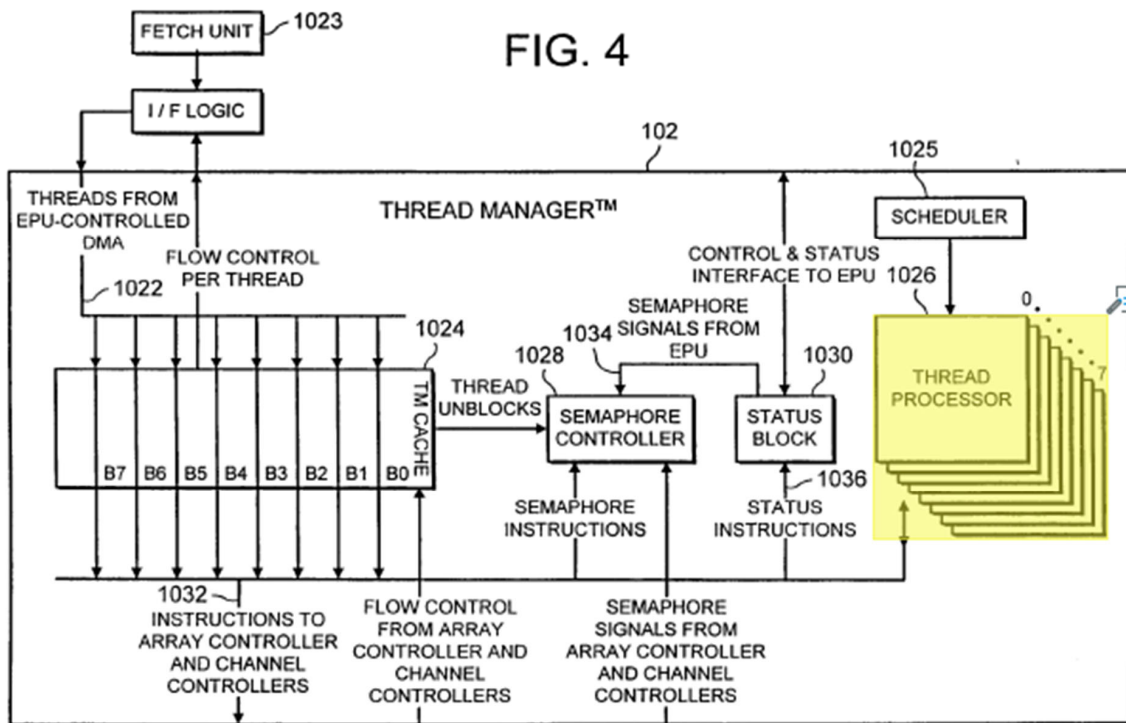
110. In my opinion, claims 15-20 are not obvious in view of Stuttard and the knowledge of a POSITA.

1. Stuttard Fails to Disclose or Render Obvious the Claimed “Shader Core” Because the Alleged Shader Core Does Not Execute a First Task While Executing a Second Task

111. Dr. Pedram opines that “Stuttard discloses a plurality of processing blocks 106, which constitutes a shader core.” Ex. 1003, ¶ 129. But neither processing blocks 106 or any other part of Stuttard can qualify as the claimed shader core because Stuttard (including in particular processing blocks 106) can only execute one instruction at a time, and over time a single thread. It cannot execute

two instructions or threads at a given time, much less two tasks. Dr. Pedram confirms that this is how Stuttard operates, opining that “[t]he status information is subsequently transferred to the thread scheduler 1025, which uses the status information to determine ***which thread should be active at any one time.***” Ex. 1003, ¶ 126; *see also id.* at ¶ 136 (“Further, Stuttard’s thread scheduler recalculates ***which thread should be active*** based in part on thread priorities.”). Dr. Pedram thus acknowledges that only one thread is active at any one time in Stuttard, and so it is impossible for the system to execute two threads (let alone two tasks) at the same time or overlapping times.

112. Stuttard uses a single “SIMD,” which as explained above in Section VI.B and VIII.C, executes the same instruction at a given time on multiple different data. Stuttard states this throughout. For example, “[a]ccording to one aspect of the present invention, there is provided a data processing apparatus comprising a SIMD (***single instruction*** multiple data) array of processing elements, wherein the processing elements are operably divided into a plurality of processing blocks, the processing blocks being operable to process respective groups of data items.” Ex. 1006 at 4:17-23. Further, “there is provided a data processing apparatus comprising an array of processing elements, which are operable to process respective data items in accordance with a common received instruction.” *Id.* at 4:25-29. Thus, the “array of processing elements” represented in Figure 3 as the multiple “Processing



Id. at Fig. 4 (annotated). Figure 4 further shows this quite clearly, as a single “thread” line leaves cache 1024 to ultimately arrive at the multiple thread processors. Similarly in Figure 3, the single thread manager sends a single instruction to all processing blocks.

113. Dr. Pedram cites to the passage in Stuttard stating “[a]t any particular moment in time, each thread may be *assigned* to any one of a number of tasks. For example, thread zero may be assigned for general system control, thread 1 assigned to execute 2D (two dimensional) activities, and threads 2 to 7 assigned to executing 3D activities (such as calculating vertices, primitives or rastering).” Ex. 1003, ¶ 129 (citing Ex. 1006 at 11:19-25). But Dr. Pedram ignores the immediately surrounding

text, which states: “In this particular example *the same instruction stream is supplied to all of the processing blocks in the system.*” Ex. 1006 at 11:33-35. At any given time, all processing blocks are executing on the same instruction.

114. Moreover, other parts of Stuttard explain that the concept of “assigning” tasks in an overlapping way does not mean they are “process[ed]” in an overlapping way, but rather only one at a time. It describes both concepts in consecutive sentences: “The processing element array can be kept active, by *processing another thread when the current active thread is halted.* The threads can be *assigned to any task as required.*” Ex. 1006 at 10:20-24. This is similar to the prior art that the ’381 patent avoided with its architecture that uses simultaneous task processing. *See* Section VII.C.

115. Furthermore, processing of a single instruction at a given time does not disclose “execute a first task from the plurality of tasks while executing a second task.” Although Stuttard *mentions* multitasking, Stuttard merely states that multithreading and multitasking are different; the former means using multiple threads, which Stuttard does (although it does not execute multiple threads at the same time, as with other prior art multi-threaded systems, see Section VI.C), and the latter means simultaneously running multiple tasks, which Stuttard states repeatedly that it does not do. Ex. 1006 at 10:7-15. As stated throughout Stuttard and explained above, Stuttard uses multiple threads, but at any given time, only one instruction is

run by all of the processing blocks in the system. The one-sentence statement of what multitasking is, while explicitly stating Stuttard is explicitly designed not to perform it, does not disclose that Stuttard meets this limitation.

2. Stuttard Fails to Disclose or Render Obvious the Claimed “Scheduling Module” and “Second Processing Unit”

a. The Alleged Scheduling Modules Are Not Associated With the Alleged Second Processing Units

116. Dr. Pedram opines that “[t]he processing core 10 is thus the claimed ‘first processing unit.’” Ex. 1003, ¶ 121. Processing core 10 is shown in Figure 1 of Stuttard:

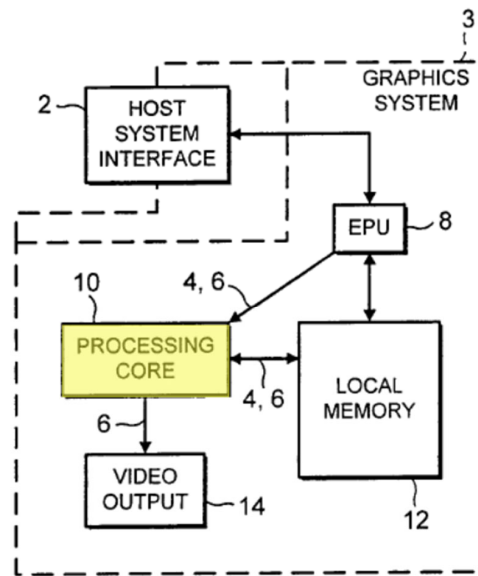


FIG. 1

Ex. 1006 at Fig. 1 (annotated).

117. Dr. Pedram proposes two theories regarding the scheduling module and the second processing unit. First, he states “this plurality of engines is configured to receive from the instruction fetch unit 1023 and/or the thread scheduler 1025 (either of which can be the claimed ‘scheduling module’) associated with the embedded processor unit EPU 8 (which is the claimed ‘second processing unit’), a plurality of tasks.”). Ex. 1003, ¶ 125. However, Stuttard expressly provides that fetch unit 1023 and thread scheduler 1025 are within the first processing unit and not the second. Ex. 1006 at 11:3-10 (“The thread manager also includes an instruction fetch unit 1023, a thread scheduler 1025, thread processors 1026, a semaphore controller 1028 and a status block 1030.”). The processing core, shown in Figure 3, demonstrates that the Thread Manager is within it:

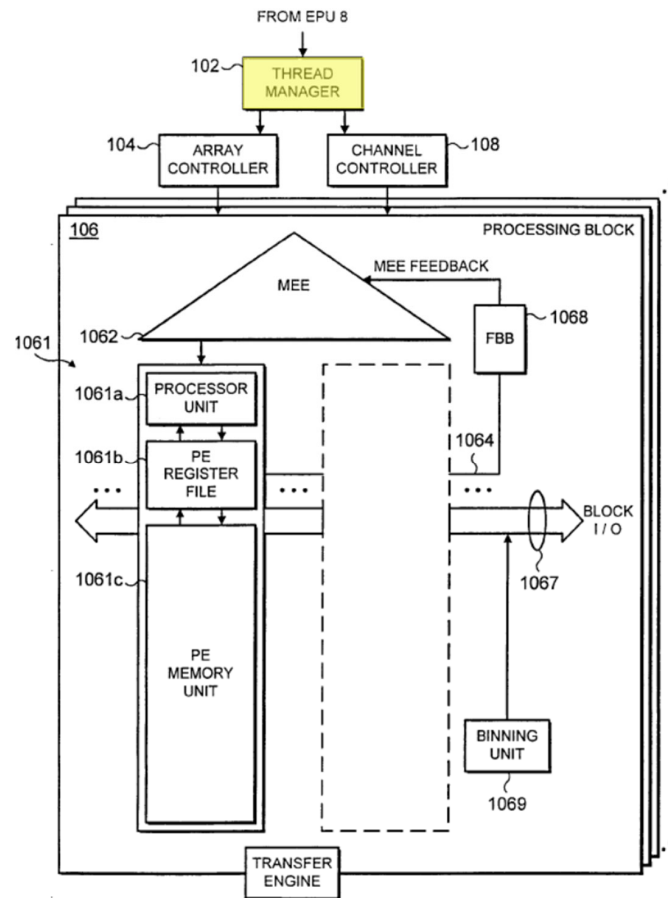
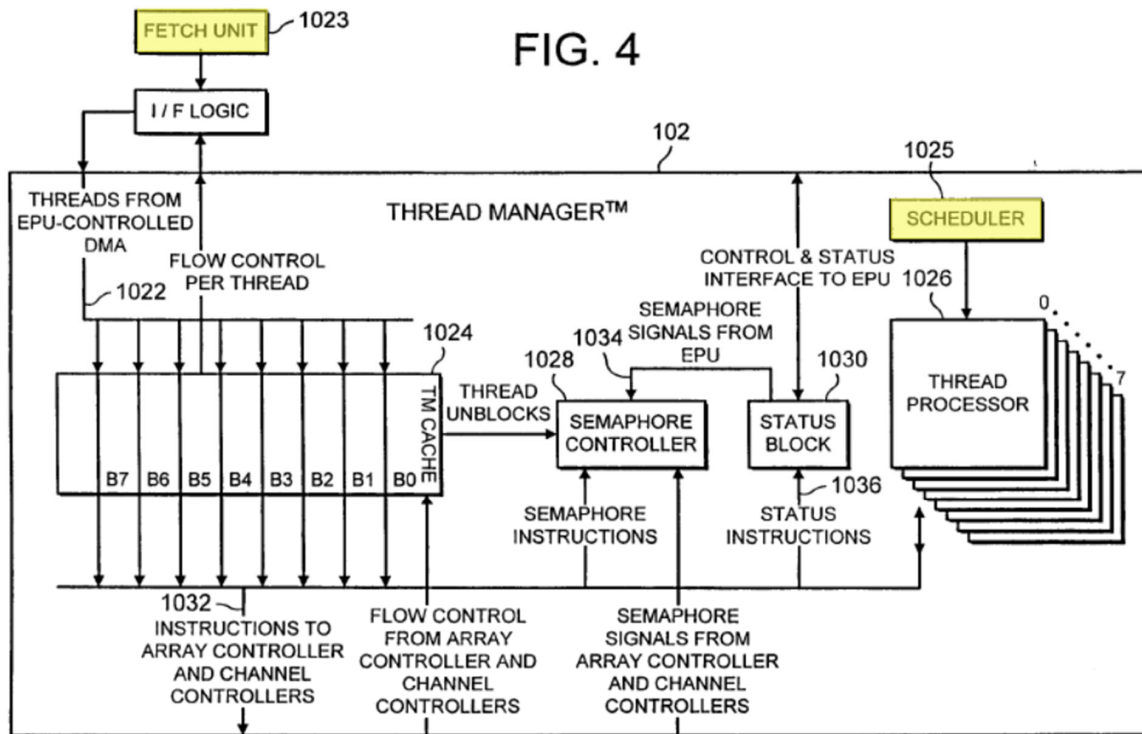


FIG. 3

Ex. 1006 at Fig. 3 (Processing Core 10) (annotated), 8:17. Figure 4, showing the Thread Manager, shows that the items alleged to be the scheduling module, Fetch Unit 1023 and Scheduler 1025, are within the thread manager and thus within the processing core 10, which is alleged to be the first processing unit.



Ex. 1006, Fig. 4. Therefore, the alleged scheduling modules are “associated with” only the first processing unit, and *not* the second processing unit, for the same reasons as explained in Section IX.B.2 above regarding Jiao and the prosecution history disclaimer regarding this term. Stuttard thus fails to disclose claim 15 under this theory. *Id.*

118. Dr. Pedram’s second theory is that “[a]lternatively, the plurality of engines is configured to receive from the EPU 8 (the claimed ‘scheduling module’) associated with a host system (the claimed ‘second processing unit’), a plurality of

tasks.” Ex. 1003, ¶ 125. EPU 8 is depicted in Figure 1, as is the interface to the host system, which is not shown:

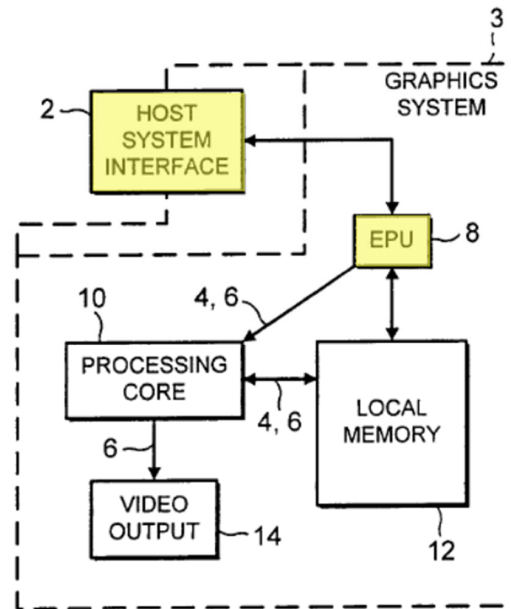


FIG. 1

Ex. 1006 at Fig. 1 (annotated), 4:21-24 (“The system 3 interfaces with a host system (not shown).”). But again, EPU 8 is not “within” the alleged second processing unit, the host, as it must be according to the prosecution history disclaimer explained above regarding Jiao in Section IX.B.2. The EPU 8 thus also fails to meet the claim requirements of a scheduling module.

b. The EPU Further Is Not a Scheduling Module Because It Does Not Transmit Tasks

119. Claim 15 requires “a plurality of engines configured to receive, from a scheduling module associated with a second processing unit, a plurality of tasks.”

Ex. 1001 at Claim 15. But EPU 8 does not transmit any tasks, and so it cannot be the claimed scheduling module. Rather, “thread manager 102” affirmatively fetches instructions for a thread (which are not tasks) when EPU 8 tells it to. Ex. 1006 at 8:17-21 (“The thread manager 102 is connected to receive control signals from the EPU 8. The control signals inform the 20 thread manager as to when instructions are to be fetched and where the instructions are to be found.”).

120. Dr. Pedram opines that “[w]hen the scheduler is disabled the EPU is able to control activation of the threads,” *i.e.*, when Scheduler 1025 is disabled, the EPU can directly “start and stop the active thread” on the alleged shader core, which is made up of processing blocks 106. Ex. 1003, ¶ 125. However, Dr. Pedram does not explain why that would constitute sending a task from EPU 8 to an alleged engine (“binning unit 1069, the channel controller 108, the array controller 104, and components of the thread manager 102 (including the cache memory unit 1024, the thread processors 1026, and the status block 1030),” *id.*), as required by the claims. Under this theory, the EPU is bypassing all of the alleged engines and interacting directly with the alleged shader core. Moreover, the deactivation of the scheduler cannot be a claimed “task,” at a minimum because the instruction to deactivate is not then received by any alleged engine or the alleged shader core, as required by the claims. Ex. 1001 at Claim 15 (“a shader core configured to receive the plurality of tasks from at least one of the plurality of engines”).

**3. Stuttard Fails to Disclose or Render Obvious the Claimed
“Plurality of Engines”**

**a. The Alleged Engines Do Not Each Receive Tasks
From The Alleged Scheduling Module**

121. As explained above in connection with Jiao, Section IX.B.1.a regarding Dr. Pedram’s alternative identification of “thread slots” as engines on pages 49-59, in claim 15, each engine must receive tasks. Dr. Pedram identifies “binning unit 1069, the channel controller 108, the array controller 104, and components of the thread manager 102 (including the cache memory unit 1024, the thread processors 1026, and the status block 1030)” as the alleged engines that receive tasks from the scheduling module. Ex. 1003, ¶ 125. Petitioner has not shown that each of its alleged engines receives a task. At a minimum the data received by Binning Unit 1069 and Status Block 1030 fail to meet even Petitioner’s broad definition of “task.” Binning Unit 1069 receives “primitive data.” Ex. 1006 at 14:32-35, 21:15-21. Primitives are basic shapes, like triangles, as explained above regarding Jiao. *See* Section IX.B.1.a. Indeed, Petitioner’s expert, Dr. Pedram, testified that the “binning unit here does not receive tasks, no,” stating “[a]bsolutely not. It does not receive tasks.” Ex. 2005 (Pedram 1/4/23 Dep. Tr.) at 284:10- 283:5-284:10. Dr. Pedram admitted that “[a] person of ordinary skill in the art would understand that the binning unit of Stuttard does not constitute an engine.” *Id.* at 288:24-289:2. And

Status Block 1030 receives “status information,” which is not a task. Ex. 1006 at 12:11-14. These alone demonstrate that Petitioner’s identification of engines fails.

122. Further, as explained below, Dr. Pedram has not shown that each of its alleged engines receives anything from the alleged scheduling modules, under either of his theories of what constitutes a scheduling module. Dr. Pedram’s first theory is that “instruction fetch unit 1023 and/or the thread scheduler 1025” are a scheduling module. Ex. 1003, ¶ 125. But of all the alleged engines, only thread processor 1026 is connected to Thread Scheduler 1025, and none of the items are connected to Fetch Unit 1023. At a minimum, it is clear that several items, such as Array Controller 104, Channel Controller 108, and Binning Unit 1069 are substantially downstream from the EPU 8, where the entirety of the Thread Manager 102, including the Cache 1024 and Thread Processor 1026 are between the EPU 8 and those components.

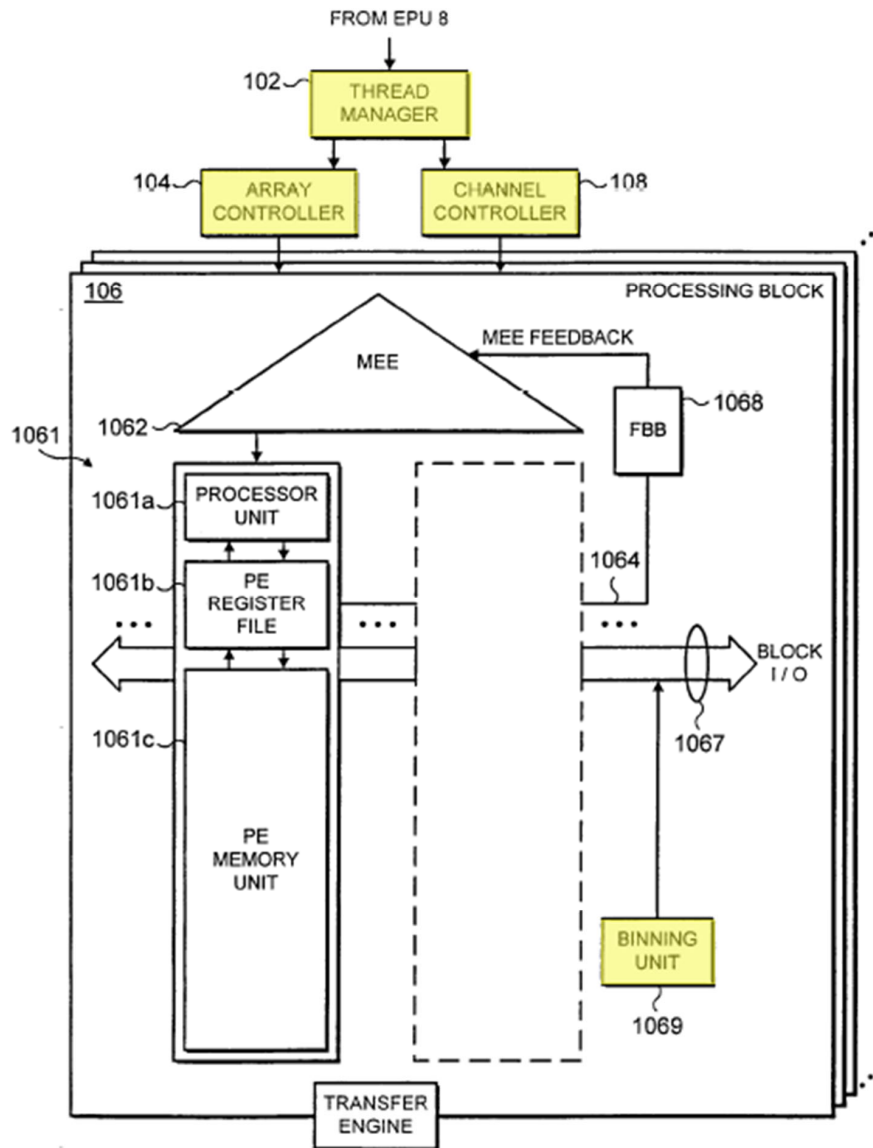
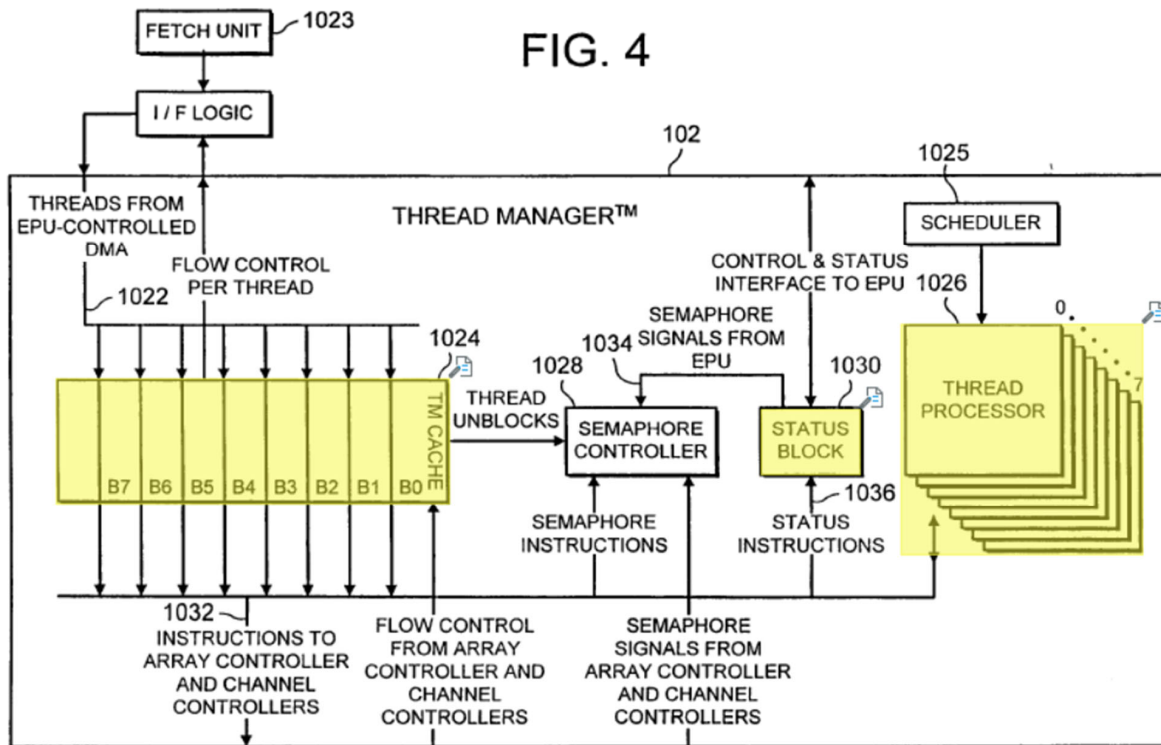


FIG. 3

Ex. 1006 at Fig. 3 (annotated).



Id. at Fig. 4 (annotated).

123. Per the interpretation Dr. Pedram, the downstream components cannot be considered to receive tasks from either of the alleged scheduling modules. *See* Section IX.A.1.b.

124. As to Dr. Pedram's second identified scheduling module (EPU 8, Ex. 1003, ¶ 125, none of the alleged engines receive anything from EPU 8, as they are not connected to EPU 8. At most, Fetch Unit 1023 is connected to EPU 8 and may receive alleged tasks from EPU 8, but Fetch Unit 1023 is not an alleged engine.

b. The Alleged Engines Do Not Each Load State Data

125. Dr. Pedram does not opine that “binning unit 1069,” “cache memory unit 1024,” or “the thread processors 1026,” load state data. Ex. 1003, ¶¶ 126-127.

X. CONCLUSION

126. 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.

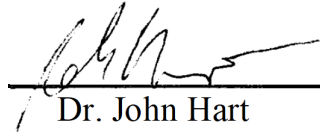
127. The statements and opinions in this declaration comprise and are based upon my review, analysis and understanding of the source materials that I have reviewed as discussed herein, based on the disclosure and statements made within such source materials.

128. 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.

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

July 28, 2023

Respectfully Submitted,



Dr. John Hart

EXHIBIT A

Curriculum Vitæ

John C. Hart

Director of Professional Education
and
Professor of Computer Science
University of Illinois at Urbana Champaign

(217) 333-8740 – jch@illinois.edu – <http://graphics.cs.illinois.edu/~jch>

∞ Experience ∞

Education

- **Ph.D.** Electrical Engineering and Computer Science Dept., University of Illinois at Chicago, 1991. Thesis title: "Computer Display of Linear Fractal Surfaces." Advisor: Thomas A. DeFanti.
- **M.S.** EECS Dept., University of Illinois at Chicago, 1989. Thesis title: "Image Space Algorithms for Visualizing Quaternion Julia Sets." Advisor: Thomas A. DeFanti.
- **B.S.** College of Liberal Arts and Science, Aurora University, 1987. Major: Computer Science.

Academic Positions

- **Director, Office of Professional Education**, University of Illinois at Urbana-Champaign, 2021-.
- **Director of Online Programs**, Dept. of Computer Science, 2018-2021.
- **Director**, Master of Computer Science in Data Science program, 2016-2017.
- **Executive Associate Dean**. Graduate College, University of Illinois, 2015-2021.
- **Associate Dean**. Graduate College, University of Illinois, 2014-2015.
- **Director for Graduate Studies**. Dept. of Computer Science. 2013.
- **Full Professor**. Dept. of Computer Science, University of Illinois, 2006-.
- **Associate Professor**. Dept. of Computer Science, University of Illinois, 2000-2006.
- **Associate Professor**. School of Electrical Engineering and Computer Science, Washington State University, 1998-2000.
- **Assistant Professor**. School of Electrical Engineering and Computer Science, Washington State University, 1992-1998.
- **Postdoctoral Research Associate**. Electronic Visualization Laboratory, University of Illinois at Chicago, and National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, 1991-1992.
- **Research Assistant**. Electronic Visualization Laboratory, Electrical Engineering and Computer Science Dept., University of Illinois at Chicago, 1988-1991.

- **Teaching Assistant.** Analog & digital circuit design courses, EECS Dept. University of Illinois at Chicago, 1987-1988.

Awards and Honors

- **The Excellence in Online & Distance Teaching Award.** University of Illinois at Urbana-Champaign, 2021.
- **Fellow.** Academic Leadership Program, Big Ten Academic Alliance, 2016-2017.
- **Outstanding Advisor.** College of Engineering, 2010 and 2015.
- **Winner.** 2004 Fantasy Graphics League.
- **Listed.** *International Who's Who of Professionals*, 1997.
- **Champion.** SIGGRAPH Bowl II, SIGGRAPH 94. Captain of the EVL Alumni Team.
- **NSF Research Initiation Award.** "Modeling, Rendering and Animation of Implicit Surfaces," to support the generalization of methods developed for the visualization of fractal models to the more common implicit surfaces used in CAGD and entertainment, 1993.
- **First Runner-Up.** Truevision Videographics Competition, SIGGRAPH 90.
- **Graduate Fellowship.** Graduate College, University of Illinois at Chicago, 1990-1991.

Industry Consulting

- **Intrinsic Medical Imaging LLC**, Bloomfield Hills, MI. Consultant, 2012-13. Mesh construction and simulation of coronary arterial blood flow.
- **Pratt & Whitney**, Hartford, CT. Consultant, 2011. Flux estimation in a lensed system.
- **Science Applications International Corp. (SAIC)**, Champaign, IL. Consultant, 2008. Ray-NURBS intersection.
- **Adobe Systems, Inc.**, Seattle, WA. Visiting Researcher, 2007. Project: Rendering meshed objects as stylized vector art.
- **The Teaching Company**, Chantilly, VA. Consultant, 2003. Produced custom educational video elements on the platonic solids.
- **Evans & Sutherland Computer Corp.**, Salt Lake City, Utah. Consultant, 1999-2000. Project: Design of the Evans & Sutherland Multi-Texturing Language.
- **Blue Sky | VIFX, Inc.**, Culver City, California. Consultant, 1998. Project: development of a custom software plug-in for the Houdini procedural animation package to polygonize a feature film villain modeled as a complex implicit surface.
- **Evans & Sutherland Computer Corp.**, (group formerly known as Silicon Reality, Inc.) Federal Way, Washington. Consultant, 1997-8. Project: design of graphics hardware to support antialiased procedural solid texturing.
- **Kleiser-Walczak Construction Company (a visual effects production company).** Lennox, Massachusetts. Consultant, 1992-1993. Project: development of a new fractal-based video transition effect for an attraction at the Luxor Hotel, Las Vegas.
- **AT&T Pixel Machines.** AT&T Bell Laboratories, Holmdel, New Jersey. Summer Intern, 1990.
- **IBM T.J. Watson Research Center.** Hawthorne, New York. Summer Intern, 1989.

Expert Testimony

(Only includes disclosed engagements.)

- **LG Electronics, Inc.** v. TCL. US District Court, Eastern District of Texas, Marshall Division. C.A. No. 2:22-cv-122-JRG. Retained by Baker Botts, LLP. Authored expert decl. on claim construction, 4/18/23. Settled.
- **Advanced Micro Devices (AMD)** v. RealTek, TCL. US International Trade Commission Case #337-TA-1318. Retained by Mintz, Levin, Cohn, Ferris, Glovsky and Popeo PC. Testified at Markman Hearing 9/6/22. Ongoing.
- **Stratos Audio.** IPR's and trials regarding Patents 8,200,203, 8,688,028 and 8,903,307. Retained by White & Case LLP. Dec. 2021-ongoing. Deposed 3/31/22, 7/8/22, 7/12/22 and 7/20/22.
- **Infernal Technologies** v. Sony Interactive Entertainment, US District Court, Eastern District of Texas Marshall Division, C.A. No. 2:19-cv-248-JRG. Retained by Beuther, Joe & Counselors. Authored expert reports on infringement (Aug. 2020) and validity (Sep. 2020). Testified in jury trial Oct. 2021.
- **Intellectual Pixels**, Inter Partes Review for Patent 8,667,093. Retained by Armond Wilson. Mar. 2021 – June 2022. Authored declaration. Deposed 18 June 2021.
- **CorePhotonics**, Inter Partes Reviews for Patents 10,225,479 and 10,230,898. Retained by Russ, August & Kabat. Jan. 2021 – ongoing. Authored declarations. Deposed 29 Apr. 2021 and 21 May 2021.
- **LG Electronics Inc.**, US District Court, Central District of California, Western Division. Case No. 2:19-cv-09474-JAK. Retained by Sidley Austin Jan. 2020. Authored Sep 2020 expert report on claim construction.
- **Align Technology, Inc.**, US District Court for the District of Delaware, C.A. No. 17-1647-LPS-CJB. Retained by Morrison Foerester Mar. 2019. Authored infringement report and reply report on validity. Deposed 19 June 2020. Settled Feb. 2022.
- **Adsync Technologies**, Middle District of Florida, Orlando Division, Case 6:18-cv-1176-Orl-40TBS. Retained by Akerman Jun. 2019. Declaration in defense of Amended Complaint from Adacel and in support of counter complaint from Adsync. Deposed 24 Jan 2020. Settled Jan. 2021.
- **ZiiLabs** (subsidiary of Creative Technology), Certain Graphics Processors and Products Containing the Same, US International Trade Commission Case #337-TA-1099. Retained by Pepper Hamilton Sep. 2017. Deposed 12 Nov. 2018.
- **Samsung.** - FotoNation Ltd. et al. v. Samsung Electronics. Co., Ltd., et al., No. 2:17-cv-00669 (E.D. Tex.). Retained by O'Melveny & Myers Oct. 2017.
- **Samsung**, Inter Partes Reviews for of patents filed by Image Processing Technologies LLC: 6,717,518, 6,959,293, 7,650,015, 8,805,001, 8,983,134 and 8,989,445. Retained by O'Melveny & Myers Oct. 2016. Deposed Dec. 2017.
- **ZiiLabs** (subsidiary of Creative Technology), Certain Graphics Processors, DDR Memory Controllers, and Products Containing the Same, US International Trade Commission Case #337-TA-1037, for Pepper Hamilton LLP, Sep. 2016 – July 2017. Testified to the International Trade Commission for Markman hearing.
- **Advanced Silicon Technology**, Certain Computing or Graphics Systems, Components Thereof, and Vehicles Containing Same, US International Trade Commission Case #337-TA-984, for Mintz, Levin, Cohn, Ferris, Glovsky and Popeo PC, Aug. 2015 – Aug. 2016. Testified to the International Trade Commission for Markman hearing and authored validity report. Written declarations submitted for IPR2016-00894, -00897 and 01060).

- **ZiiLabs** (subsidiary of Creative Technology), Eastern District of Texas, Marshall Division, Case 2:14-cv-00203, for Heim, Payne & Chorush LLP and Susman Godfrey LLP. Sep. 2014 – Dec. 2015. Prepared invalidity rebuttal report for four patents and damages report for nine patents, all on graphics hardware. Deposed over two sequential days on both reports.
- **Graphics Property Holdings** (formerly SGI), Certain Consumer Electronics with Display and Processing Capabilities, US International Trade Commission Case #337-TA-884, for Mintz, Levin, Cohn, Ferris, Glovsky and Popeo . Graphics hardware expert on floating point rasterization, Authored report on patent validity. Deposed Feb. 2014. Testified to the International Trade Commission Apr. 2014.
- **Microsoft Xbox**, Illinois Northern District Court (Eastern Division) Case #08-C-393, for Sidley Austin LLP. Dec. 2009 – May 2010. Prepared non-infringement rebuttal report and invalidity report.
- **NVIDIA & AMD (ATI)**, California Northern District Court Case # 3:07-cv-00302, for Cooley Godward Kronish LLP. July 2008 – Sep. 2008. Provided expertise on graphics hardware and insight into the industry-wide process for the deployment of innovative features.
- **Broadcom**, California Southern District Court Case #06-CV-0660, for McAndrews, Held & Malloy, Ltd, Dec. 2006 – Mar. 2007.

Research Funding

From Industry

- **Caterpillar**, through the **Digital Manufacturing & Design Innovation Institute**. \$410,000, Co-PI with Placid Ferreira and Shiv Kapoor.
- **Intel**. Intel Illinois Parallelism Center. \$2,500,000, Co-PI, 2011-2012.
- **Intel**. Leveraging Larrabee's Programmable Rasterization. \$130,000, 2009-2011.
- **Intel/Microsoft**. Universal Parallel Computing Research Center. \$18,000,000, Co-PI, 2008-2012.
- **NAVTEQ**. "Surface Classification and Reconstruction from LIDAR and images." \$60,000, 2008.
- **Thomas M. Siebel**. "MethMorph: Visual Simulation of Methamphetamine Addiction." \$35,000, Spring 2006.
- **Nvidia Corp.** \$15,000 annually through the UIUC CS Affiliates Program, 2004–, and \$125,000 total to date in Ph.D. student fellowships (\$25K directly to the student to support one year of Ph.D. research): Nate Carr: 2002 & 2003, Jesse Hall: 2003 & 2004, Jared Hoberock: 2005 & 2008.
- **Microsoft Research**. "Precomputed Radiance Transfer Compression." \$15,000, Sep. 2002.
- **Evans & Sutherland Computer Corp.** "Real Time Procedural Solid Texturing." \$83,246, June 1999 – May 2000.
- **Evans & Sutherland Computer Corp. and Washington Technology Center**. "APST: Computer Graphics Hardware for Antialiased Procedural Solid Texturing." \$77,089. Aug. 16, 1998 - Aug. 15, 1999.
- **Intel Natural Data Types Group**. "Procedural Modeling." \$43,000. Aug. 16, 1997 - Aug. 15, 1998.
- **Intel Natural Data Types Group**. "Recurrent Modeling -- Beyond Fractal Block Coding." (Co-PI with P. Flynn.) \$93,000. Apr. 1, 1995 - Mar. 31, 1997.

From Government Agencies

- **NSF. #OAC-1550554** “SI2-SSI: Collaborative Research: ParaTreet: Parallel Software for Spatial Trees in Simulation & Analysis” (co-PI with Laxmikant Kale), \$170,000, Sep. 2016 – Aug. 2018.
- **NSF. #OCI-1216788** “Collaborative Research: Conceptualizing an Institute for Using Inter-Domain Abstractions to Support Inter-Disciplinary Applications” (co-PI with David Padua and Philippe Geubelle, and other collaborators at Purdue and UT-Austin), \$135,000, Oct. 2012 – Sep. 2016.
- **NSF. #EF-1115112** “Collaborative Research: Digitization TCN: InvertNet--An Integrative Platform for Research on Environmental Change, Species Discovery and Identification” (co-PI with Christopher Dietrich, Christopher Taylor, Nahil Sobh and Umberto Ravaioli), \$2,809,463, July 2011-June 2015.
- **NSF. #OCI-1047764** “SI2-SSE: Collaborative Research: Lagrangian Coherent Structures for Accurate Flow Structure Analysis” (co-PI with Shawn Shadden) \$251,643, Sep. 2010 – Aug. 2013.
- **NSF. #IIS-0534485** “Analysis and Visualization of Complex Graphs” (co-PI with Michael Garland) \$300,000, Sep. 2006 – Aug. 2009.
- **UIUC Critical Research Initiative.** “A New Approach to Bone Replacement.” (Co-PI with Russ Jamison, Michael Goldwasser, Ben Grosser, Matei Stroila and Amy Wagoner Johnson.) \$100,000, Sep. 2005 – Aug. 2006.
- **NSF Small Grant for Exploratory Research.** “Application Directed Surface Parameterization.” \$97,868, Jan. 2005 – Dec. 2005.
- **NSF/DARPA CARGO (Computational and Algorithmic Representations of Geometric Objects) Award.** “Robust Lagrangian Surface Propagation with Topological Control.” (Lead PI, with Michael Heath, Jim Jiao and John Sullivan), \$400,000, May 2003 – May 2006.
- **NSF Information Technology Research. #NSG-0219594** “Making 3D Visibility Practical.” (Co-PI with Steve LaValle, Jeff Erickson, Fredo Durand) \$499,923. Aug. 2002 – Aug. 2005.
- **CNRS (Centre National de la Recherche Scientifique).** Supplement to “Making 3D Visibility Practical” to support UIUC – INRIA collaboration, \$7,000, 2003-5.
- **NSF Information Technology Research. #ACI-0113968** “Multipass Programming for Personal High-Performance Computing.” \$489,671, Aug. 2001 – July 2006.
- **NSF Information Technology Research. #ACI-0121288** “Procedural Representation and Visualization Enabling Personalized Computational Fluid Dynamics.” (Co-PI with David Ebert, David Marcum, Kelly Gaither and Penny Rheingans) \$3,989,773. Aug. 2001 – July 2006.
- **NSF. #NSG-9732379** “Applications of Morse Theory and Catastrophe Theory to Computer Graphics.” (Co-PI with R. Lewis.) \$220,541. Aug. 16, 1997 – Aug. 15 2000.
- **NSF. #CCR-9529809** “Recurrent Modeling.” (Co-PI with P. Flynn.) \$206,435. June 15, 1996 - May 31, 1999. Research Experience for Undergraduates Supplement: \$5,000. Jan. 1, 1998 - May. 31, 1999.
- **NSF Research Initiation Award. #CCR-9309210** “Modeling, Rendering and Animation of Implicit Surfaces.” \$97,925. July 1, 1993 - June 30, 1996. Research Experience for Undergraduates Supplement: \$4,885. Jan. 1, 1995 - Dec. 31, 1995.

Education Projects

- **Nokia University Cooperation Funding.** “Teaching Mobile Augmented Reality on the Windows Phone Platform.” \$11,377.64. 2013.

- **UIUC College of Engineering Strategic Instructional Initiatives Program.** Improvement of key ME courses (Statics, Dynamics and Solid Mechanics), \$450,000, with MechSE Profs. Tortorelli, Dullerud Keane and West, 2012-2014.
- **NSF Special Project.** #EIA-9911033 “The Story of Computer Graphics Documentary Project.” \$48,000. Sept. 15, 1999 – Aug. 31, 2000.

Equipment Grants and Donations

- **Nokia.** Two Velodyne LIDAR scanners, 2013.
- **NVidia.** K20 graphics card, 2012.
- **Intel.** Dell XPS 12, \$1,000, 2012.
- **Intel.** Two Knights Ferry development workstations, \$4,940, 2009.
- **NVidia.** Various graphics cards, 2002-4.
- **ATI.** One graphics card, 2002.
- **Tektronix Phaser 340 Color Printer.** Tektronix. \$5,000. Oct. 1995.
- **High-Performance Networking and Computing Infrastructure for Imaging Research.** (Co-PI with T. Fischer, P. Flynn and R. Bamberger.) NSF Research Instrumentation and Infrastructure. CISE Instrumentation Program, Office of Cross-Disciplinary Activities. \$100,000. March 1, 1994 - April 30, 1996. Awarded: Jan. 1995.
- **Silicon Graphics Inc. Workstation Upgrade.** \$9,000. Apr. 1993.
- **XAOS Tools.** Software Donation. Pandemonium and n-title. \$6,500. Feb. 1993.
- **Karen Guzak.** Art Donation. 12 Prints. \$4,800. May 1995.

Publications

Most of the following publications are available online via
<http://graphics.cs.illinois.edu/~jch/papers>.

Reviewed Publications

1. A.I. Ellis, W. Hunt and J.C. Hart. Real-Time Analytic Antialiased Text for 3-D Environments. Proc. High-Performance Graphics, Computer Graphics Forum 38(8), 2019, pp. 23-32.
2. A.J. Christensen, V. Srinivasan, J.C. Hart and A. Marshall-Colon. Use of computational modeling combined with advanced visualization to develop strategies for the design of crop ideotypes to address food security. Nutrition Reviews 76(5), May 2018, pp. 332-347.
3. P-C Wang, A.I. Ellis, J.C. Hart, C-H Hsu. Optimizing next-generation cloud gaming platforms with planar map streaming and distributed rendering. Proc. NetGames (15th Annual Workshop on Network and Systems Support for Games), 2017.
4. A. Marshall-Colon, S.P. Long, D.K. Allen, G. Allen, D.A. Beard, B. Benes, S. von Caemmerer, A.J. Christensen, D.J. Cox, J.C. Hart, P.M. Hirst, K. Kannan, D.S. Katz, J.P. Lynch, A.J. Millar, B. Panneerselvam, N.D. Price, P. Prusinkiewicz, D. Raila, R.G. Shekar, S. Shrivastava, D. Shukla, V. Srinivasan, M. Stitt, M. Turk, E.O. Voit, Y. Wang, X. Yin, X. Zhu. Crops in silico: Generating Virtual Crops Using an Integrative and Multi-scale Modeling Platform. Frontiers in Plant Science, May 2017.

5. M. Chen, J.C. Hart and S. Shadden. Hierarchical Watershed Ridges for Visualizing Lagrangian Coherent Structures. Reviewed, accepted and presented at TopoInVis: Topology-Based Methods in Visualization, 2015. Published in: Topological Methods in Data Analysis and Visualization IV, H. Carr, C. Garth, T. Weinkauff, Eds. Springer Mathematics and Visualization Series, 2017, pp. 237-252.
6. A.I. Ellis, Warren Hunt and J.C. Hart. "SVGPU: Real Time 3D Rendering to Vector Graphics Formats." Proc. High Performance Graphics, 2016.
7. M. Chen, S. Shadden and J.C. Hart. "Fast Coherent Particle Advection through Time-Varying Unstructured Flow Datasets." IEEE Transactions on Visualization and Computer Graphics 22(8), pp. 1959-1972, 2016.
8. J. Kratt, M. Spicker, A. Guayaquil, M. Fiser, S. Pirk, O. Deussen, J.C. Hart, B. Benes. Woodification: User-Controlled Cambial Growth Modeling. Proc. Eurographics, Computer Graphics Forum 23(2), May 2015, pp. 361-372.
9. V. Lu, J.C. Hart. Multicore Construction of k-d Trees for High Dimensional Point Data. Proc. Advances in Big Data Analytics, July 2014.
10. P.R. Khorrami, V.V. Le, J.C. Hart, T.S. Huang. A System for Monitoring the Engagement of Remote Online Students using Eye Gaze Estimation. Proc. IEEE ICME Workshop on Emerging Multimedia Systems and Applications, July 2014.
11. V. Lu, I. Endres, M. Stroila and J.C. Hart. Accelerating Arrays of Linear Classifiers Using Approximate Range Queries. Proc. Winter Conference on Applications of Computer Vision, Mar. 2014.
12. D. Mayerich, J.C. Hart, Volume Visualization of Serial Electron Microscopy Images Using Local Variance. Proc. BioVis (IEEE Symposium on Biological Data Visualization), pp. 9-16, Oct. 2012
13. C. Dietrich, J. Hart, D. Raila, U. Ravaioli, N. Sobh, O. Sobh, C. Taylor. InvertNet: a new paradigm for digital access to invertebrate collections. ZooKeys 209, July 2012, pp. 165-181.
14. Y. Jia, M. Garland, and J.C. Hart. Social Network Clustering and Visualization using Hierarchical Edge Bundles. Computer Graphics Forum 30(8), Dec. 2011, pp. 2314-2327.
15. Y. Jia, V. Lu, J. Hoberock, M.J. Garland, J.C. Hart. Edge v. Node Parallelism for Graph Centrality Metrics. GPU Computing Gems – Jade Edition, Oct. 2011, pp 15-28.
16. M. Kamali, F.N. Iandola, H. Fang, J.C. Hart. MethMorph: Simulating Facial Deformation due to Methamphetamine Usage. Proc. International Symposium on Visual Computing. Sep. 2011.
17. M. Kamali, J. Cho, M. Stroila, E. Shaffer, J.C. Hart. Robust Classification of Curvilinear and Surface-like Structures in 3D Point Cloud Data. Proc. International Symposium on Visual Computing. Sep. 2011.
18. M. Kamali, E. Ofek, F. Iandola, I. Omer, J.C. Hart. Linear Clutter Removal from Urban Panoramas. Proc. International Symposium on Visual Computing. Sep. 2011.
19. K. Karsch, J.C. Hart. Snaxels on a Plane. Proc. Non-Photorealistic and Artistic Rendering, Aug. 2011.
20. J. Hoberock, J.C. Hart. Arbitrary Importance Functions for Metropolis Light Transport. Computer Graphics Forum 29(6), 2010, pp. 1993-2003.
21. S. Shi, M. Kamali, K. Nahrstedt, J.C. Hart, R. Campbell. High-Quality Zero-Delay Remote Rendering System for 3D Video. Proc. Multimedia, Oct. 2010.
22. B. Choi, R. Komuravelli, V. Lu, H. Sung, R.L. Bochino, S.V. Adve, J.C. Hart. Parallel SAH k-D Tree Construction. Proc. High Performance Graphics, June. 2010.
23. Y. Jia, X. Ni, E. Lorimer, M. Mullan, R. Whitaker, J.C. Hart. RBF Dipole Surface Evolution. Proc. Shape Modeling International, June 2010.
24. W.-W. Feng, B.-U. Kim, Y. Yu, L. Peng, J.C. Hart, Feature-Preserving Triangular Geometry Images for Level-of-Detail Representation of Static and Skinned Meshes, ACM Transactions on Graphics 29(2), March 2010, Article 11.

25. J. Hoberock, V. Lu, Y. Jia, J.C. Hart. Stream Compaction for Deferred Shading. Proc. High Performance Graphics, Aug. 2009.
26. J.C Hart. Assistive Technology for the Aesthetically Impaired. Proc. CHI Workshop on Computational Creativity Support, Apr. 2009.
27. J. Hoberock, S. Hornus, J.C. Hart. On Constructing and Visualizing the Topological Structure of the Visibility and Radiance of Architectural Models. Proc. TopoInVis: Topological Methods in Data Analysis and Visualization, Feb. 2009.
28. Y. Jia, J. Hoberock, M. Garland, J.C. Hart. On the Visualization of Social and other Scale-Free Networks. (Proc. Infovis), IEEE Transactions on Visualization and Computer Graphics 14(6), Nov. 2008, pp. 1285-1292.
29. A. Godiyal, J. Hoberock, M. Garland, J.C. Hart. Rapid Multipole Graph Drawing on the GPU. Proc. Graph Drawing, Sep. 2008.
30. E. Eisemann, H. Winnemoller, J.C. Hart, D. Salesin. Stylized Vector Art from 3D Models with Region Support. (Proc. Eurographics Rendering Symposium), Computer Graphics Forum 27(4), June 2008, pp. 1199-1207.
31. M. Stroila, E. Eisemann, J.C. Hart. Clip Art Rendering of Smooth Isosurfaces. IEEE Transactions on Visualization and Computer Graphics 14(1), Jan. 2008, pp. 71-81.
32. H. Fang, J.C. Hart. Detail Preserving Shape Deformation in Image Editing. Proc. SIGGRAPH, ACM Transactions on Graphics 26(3), Aug. 2007, #12.
33. T. Bergstrom, K. Karahalios, J.C. Hart. Isochords: Visualizing Structure in Music. Proc. Graphics Interface, May 2007.
34. H. Fang, J.C. Hart. RotoTexture: Automated Tools for Texturing Raw Video. IEEE Transactions on Visualization and Computer Graphics 12(6), Nov. 2006, pp. 1580-1589.
35. X. Jiao, A. Colombi, X. Ni, J.C. Hart. Anisotropic Mesh Adaptation for Evolving Triangulated Surfaces. Proc. Meshing Roundtable, Sept. 2006, pp. 173-190.
36. S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, J.C. Hart. Spectral Surface Quadrangulation. Proc. SIGGRAPH, ACM Transactions on Graphics 25(3), July 2006, pp. 1057-1066.
37. N.A. Carr, J. Hoberock, K. Crane, J.C. Hart. Rectangular Multi-Chart Geometry Images. Proc. Sym. on Geom. Proc., July 2006, pp. 181-190.
38. N.A. Carr, J. Hoberock, K. Crane, J.C. Hart. Fast GPU Ray Tracing of Dynamic Meshes using Geometry Images. Proc. Graphics Interface, May 2006, pp. 203-209.
39. P.-T. Bremer, J.C. Hart. A Sampling Theorem for MLS Surfaces. Proc. Point Based Graphics, June 2005.
40. J.O. Talton, N.A. Carr, J.C. Hart. Voronoi Rasterization of Sparse Point Sets. Proc. Point Based Graphics, June 2005.
41. W. Su, J.C. Hart, A Programmable Particle System Framework for Shape Modeling, Proc. Shape Modeling International, June 2005.
42. S. Zelinka, H. Fang, M. Garland, J.C. Hart. Interactive Material Replacement in Photographs. Proc. Graphics Interface, May 2005.
43. S. Hornus, J. Hoberock, S. Lefebvre and J.C. Hart. ZP+: Correct Z-pass Stencil Shadows. Proc. ACM Symposium on Interactive 3-D Graphics and Games, Apr. 2005. **(Highest ranked paper according to review score.)**
44. H. Fang, J.C. Hart, "Textureshop: Texture Synthesis as a Photograph Editing Tool," ACM Transactions on Graphics 23(3) (also in Proc. SIGGRAPH 2004), Aug. 2004, pp. 354-359.
45. X. Ni, M. Garland, J.C. Hart, "Fair Morse Functions for Extracting the Topological Structure of a Surface Mesh," ACM Transactions on Graphics 23(3) (also in Proc. SIGGRAPH 2004), Aug. 2004, pp. 613-622. **(Second highest ranked paper according to tertiary review score.)**

46. N. Carr, J.C. Hart, "Painting Detail," ACM Transactions on Graphics 23(3) (also in Proc. SIGGRAPH 2004), Aug. 2004, pp. 845-852.
47. N. Carr, J.C. Hart, "Dynamic Clustering of Meshed Surfaces," Proc. Symposium on Geometry Processing, July 2004.
48. O.S. Lawlor, J.C. Hart, "Bounding Recursive Procedural Models using Convex Optimization," Proceedings of Pacific Graphics 2003, October 2003, pp. 283-292.
49. P-P. Sloan, J. Hall, J. Hart, J. Snyder. Clustered Principal Components for Precomputed Radiance Transfer. (Proc. SIGGRAPH 2003) ACM Transactions on Graphics 22(3), July 2003.
50. N.A. Carr, J.D. Hall and J.C. Hart, GPU Algorithms for Radiosity and Subsurface Scattering, Proc. Graphics Hardware, July 2003.
51. John C. Hart, Jeyprakash Michaelraj, Brent Baker. Structural Simulation of Tree Growth. The Visual Computer 19(2-3), May 2003, pp. 151-163.
52. P. Sherman, J.C. Hart. Direct Manipulation of Recurrent Models. Computers & Graphics 27(1), Feb. 2003, pp. 143-151.
53. A. Sheffer, J.C. Hart. Seamster: Inconspicuous Low-Distortion Texture Seam Layout. Proc. Visualization 2002, Oct. 2002.
54. N.A. Carr, J.D. Hall and J.C. Hart. The Ray Engine. Proc. Graphics Hardware 2002, Sep. 2002.
55. H. Fang, J. Hart. Randomly Accessible Procedural Animation of Physically Approximate Turbulent Motion. Proc. Computer Animation 2002, June 2002.
56. J. Hart, E. Bachta, W. Jarosz, T. Fleury. Using Particles to Sample and Control More Complex Implicit Surfaces. Proc. Shape Modeling International 2002, May 2002, pp. 129-136.
57. N. Carr, J. Hart. Meshed Atlases for Real-Time Procedural Solid Texturing. ACM Transactions on Graphics 21(2), Apr. 2002.
58. J.C. Hart. Perlin Noise Pixel Shaders. *Proc. Graphics Hardware 2001*, Aug. 2001.
59. W. Cochran, R. Lewis, J. Hart. The Normal of a Fractal Surface. *The Visual Computer* 17(4), 2001, pp. 209-218.
60. J.C. Hart. Using the CW-Complex to Represent the Topological Structure of Implicit Surfaces and Solids. *Proc. Implicit Surfaces '99*, Sept. 1999, pp. 107-112.
61. J.C. Hart, N. Carr, M. Kameya, S.A. Tibbitts, T.J. Coleman. Antialiased Parameterized Solid Texturing Simplified for Consumer-Level Hardware Implementation. *Proc. Graphics Hardware 99*, Aug. 1999, pp. 45-53.
62. J.C. Hart, A. Durr and D. Harsch. Critical points of polynomial metaballs. *Proc. Implicit Surfaces '98*, June 1998, pp. 69-76.
63. J.C. Hart. Morse Theory for Implicit Surface Modeling. In *Mathematical Visualization*, H-C Hege and K. Polthier, Eds., Springer-Verlag, Berlin, Oct. 1998, pp. 257-268. (Reviewed paper presented at VisMath '97, Berlin, 1997.)
64. W.O. Cochran, J.C. Hart, P.J. Flynn. On Approximating Rough Curves with Fractal Functions. *Proc. Graphics Interface '98*, June 1998, pp. 65-72.
65. A.K. Dunker, E. Garner, S. Guilliot, P. Romero, K. Albrecht, J. Hart, Z. Obradovic, C. Kissinger and J.E. Villafranca. Protein Disorder and the Evolution of Molecular Recognition: Theory, Predictions and Observations. *Pacific Symposium on Biocomputing* 3: 435-446, Jan. 1998.
66. B. Stander, J.C. Hart. Guaranteeing the Topology of an Implicit Surface Polygonization. *Computer Graphics Annual Conference Series* (Proc. SIGGRAPH 97) Aug., 1997, pp. 279-286.
67. J.C. Hart. Implicit formulations of rough surfaces. *Computer Graphics Forum* 16(2), June 1997, pp. 91-99.
68. B. Burch, J.C. Hart. Linear Fractal Shape Interpolation. *Proc. Graphics Interface '97*. May, 1997, pp. 155-162.

69. J.C. Hart, P.J. Flynn, W.O. Cochran. Similarity Hashing: A model-based vision solution to the inverse problem of recurrent iterated function systems. *Fractals* 5, Apr. 1997, pp. 39-50.
70. W.O. Cochran, J.C. Hart, P.J. Flynn. Fractal volume compression. *IEEE Transactions on Visualization and Computer Graphics* 2(4) Dec. 1996, pp. 313-322.
71. J.C. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12(10), Dec. 1996, pp. 527-545.
72. S. Worley, J.C. Hart. Hyper-rendering of hyper-textured surfaces. Proc. of *Implicit Surfaces '96*, Oct. 1996, pp. 99-104.
73. J.C. Hart, B. Baker. Implicit Modeling of Tree Surfaces. Proc. of *Implicit Surfaces '96*, Oct. 1996. pp. 143-152.
74. J.C. Hart. Fractal image compression and the inverse problem of recurrent iterated function systems. *IEEE Computer Graphics and Applications* 16(4) July 1996, pp. 25-33.
75. K. Albrecht, J. Hart, A. Shaw, and A.K. Dunker. Quaternion contact ribbons: A new tool for visualizing intra- and intermolecular interactions in proteins. *Pacific Symposium on Biocomputing I*: 41-52, Jan. 1996.
76. D. Hepting, J.C. Hart. The escape buffer: Efficient computation of escape time for linear fractals. Proc. of *Graphics Interface '95*, May 1995. pp. 204-214.
77. J.C. Hart. Implicit formulations of rough surfaces. Proc. *Implicit Surfaces '95*, Apr. 1995, pp. 33-44.
78. B.T. Stander, J.C. Hart. A Lipschitz method for accelerated volume rendering. Proc. of the *1994 Symposium on Volume Visualization*, Oct. 1994. pp. 107-114.
79. J.C. Hart. Distance to an Ellipsoid. Chapter in: *Graphics Gems IV*, P. Heckbert, Ed., Academic Press, 1994, pp. 113-119.
80. J.C. Hart, G.K. Francis, L.H. Kauffman. Visualizing quaternion rotation. *ACM Transactions on Graphics* 13(3) July 1994, pp. 256-276.
81. J.C. Hart. On recording virtual environments. Proc. of *IEEE Visualization '93 Symposium on Research Frontiers in Virtual Reality*, Nov. 1993, pp. 80-83.
82. J.C. Hart, G.W. Lescinsky, D.J. Sandin, T.A. DeFanti, L.H. Kauffman. Scientific and artistic investigation of multidimensional fractals on the AT&T Pixel Machine. *Visual Computer* 7(9) July 1993, pp. 346-355.
83. C. Cruz-Niera, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, J.C. Hart. The CAVE audio visual experience automatic virtual environment. *Communications of the ACM* 35(6) June 1992, pp. 64-72.
84. J.C. Hart. The object instancing paradigm for linear fractal modeling. Proc. of *Graphics Interface '92*, Morgan Kaufman, May 1992, pp. 224-231.
85. J.C. Hart, T.A. DeFanti. Efficient antialiased rendering of 3-D linear fractals. *Computer Graphics* 25(3) (Proc. SIGGRAPH 91) Aug. 1991, pp. 91-100.
86. J.C. Hart, L.H. Kauffman, D.J. Sandin. Interactive visualization of quaternion Julia sets. Proc. of *Visualization '90*. IEEE Computer Society Press. Oct. 1990, pp. 209-218.
87. J.C. Hart, A. Norton. Use of curves in rendering fractures. Proc. of *SPIE/SPSE Symposium on Curves and Surfaces in Computer Vision and Graphics*. Feb. 1990.
88. J.C. Hart, D.J. Sandin, L.H. Kauffman. Ray tracing deterministic 3-D fractals. *Computer Graphics* 23(3) (Proc. SIGGRAPH 89) Aug. 1989, pp. 289-296.

Books, Invited Book Chapters and Edited Course Notes

89. J. Torrellas, S.V. Adve, V.S. Adve, D. Dig, M.N. Do, M.J. Garzaran, J.C. Hart, T.S. Huang, W.W. Hwu, S.T. King, D. Marinov, K. Nahrstedt, D.A. Padua, M. Parthasarathy, S.J. Patel, M. Snir. "Making Parallel Programming Easy: Research Contributions from Illinois," Sep. 2013.

90. M. Olano, W. Heidrich, J. Hart and M. McCool, Real Time Shading, AK Peters, July 2002.
91. J.C. Hart, "Procedural Geometry," Chapter in Modeling and Texturing, a Procedural Approach, 3rd Edition, D. Ebert, Editor, Morgan Kauffman, November 2002.
92. J.C. Hart, "Real-Time Procedural Solid Texturing," Chapter in Modeling and Texturing, a Procedural Approach, 3rd Edition, D. Ebert, Editor, Morgan Kauffman, November 2002.
93. D. Ebert and J. Hart, Editors, "Procedural Implicit Techniques for Modeling and Texturing," ACM SIGGRAPH 1998 Course Notes, July 1998.
94. D. Ebert and J. Hart, Editors, "New Frontiers in Modeling and Texturing," ACM SIGGRAPH 1997 Course Notes, August 1997.
95. J. Hart and D. Saupe, Editors, "Fractal Models for Image Synthesis, Compression and Analysis," ACM SIGGRAPH 1996 Course Notes, August 1996.
96. J. Hart, Editor, "New Directions for Fractal Modeling in Computer Graphics," ACM SIGGRAPH 1994 Course Notes, July, 1994.
97. J. Hart and K. Musgrave, Editors, "Fractal Modeling in 3-D Computer Graphics and Imaging," ACM SIGGRAPH 1991 Course Notes, July 1991.

Patents

98. H. Fang and J.C. Hart, Methods and Systems for Image Modification. #7,365,744. Filed: 26 July 2004, Granted: 29 April 2008.
99. S. Tibbitts, T. Coleman and J.C. Hart. Antialiased Procedural Solid Texturing Hardware. Provisional Patent #60,088,879. Evans & Sutherland, 1998.

Invited Papers, Articles, Tech Reports and Other Publications

100. S.V. Adve, V.S. Adve, G. Agha, M.I. Frank, M.J. Garzaran, J.C. Hart, W.W. Hwu, R.E. Johnson, L.V. Kale, R. Kumar, D. Marinov, K. Nahrstedt, D. Padua, M. Parthasarathy, S.J. Patel, G. Rosu, D. Roth, M. Snir, J. Torrellas, C. Zilles. Parallel Computing Research at Illinois: The UPCRC Agenda. Whitepaper, Nov. 2008.
101. S. Dong, P.-T. Bremer, M. Garland, V. Pascucci and J.C. Hart. Quadrangulating a Mesh using Laplacian Eigenvectors. Tech Rep. UIUCDCS-R-2005-2583, June 2005.
102. P. Lacz and J.C. Hart. Procedural Geometric Synthesis on the GPU. Poster (with accompanying manuscript) for SIGGRAPH 2004 and GP2: The ACM Workshop on General Purpose Computing on Graphics Processors, Aug. 2004.
103. J.D. Hall and J.C. Hart. GPU Acceleration of Iterative Clustering. Poster (with accompanying manuscript) for SIGGRAPH 2004 and GP2: The ACM Workshop on General Purpose Computing on Graphics Processors, Aug. 2004.
104. M. Mullan, R. Whitaker and J.C. Hart. Procedural Level Sets. Manuscript presented at NSF/DARPA CARGO Annual Meeting, May 2004.
105. J.D. Hall, N.A. Carr and J.C. Hart. Cache and Bandwidth Aware Matrix Multiplication on the GPU. Tech. Report UIUCDCS-R-2003-2328, May 2003.
106. M. Kameya, and J.C. Hart. Bresenham noise. SIGGRAPH 2000 Conference Abstracts and Applications, July 2000.
107. J.J. Kim and J.C. Hart. Contour-Based Polygonization of Regular Grid Terrain Data. Proc. Western Computer Graphics Symposium. Mar. 1999.

- 108.A. Aceves, K. Hammil and J.C. Hart. Proposal and Partially Implemented: The Studio: A Large High Resolution Immersive Projection System for Research and Education. Proc. Western Computer Graphics Symposium. Mar. 1999.
- 109.M. Kameya and J.C. Hart. Bit Width Necessary for Solid Texturing Hardware. Proc. Western Computer Graphics Symposium. Mar. 1999.
- 110.N. Carr and J.C. Hart. APST Antialiased Procedural Texturing Interface for OpenGL. Proc. Western Computer Graphics Symposium. Mar. 1999.
- 111.C. Kulratanayan and J.C. Hart. Restructuring the Scene Graph for Efficient Visibility of Natural Scenes. Proc. Western Computer Graphics Symposium. Mar. 1999.
- 112.J.C. Hart. Computational Topology for Shape Modeling. Invited Talk and Paper. Proc. Shape Modeling International '99, Univ. Aizu, Japan, Mar. 1999, pp. 36-45.
- 113.J.C. Hart. Morse Theory for Computer Graphics. WSU Technical Report EECS-97-002. Chapter in ``New Frontiers in Modeling and Texturing," SIGGRAPH '97 course notes, J. Hart, D. Ebert eds., Aug. 1997.
- 114.I.M. Danciu, J.C. Hart. Fractal Color Image Compression in the $L^*a^*b^*$ Uniform Color Space. Poster. Abstract in: Proc. of Data Compression Conference '97. IEEE Computer Society Press, Apr. 1997, p. 434.
- 115.B.T. Stander, J.C. Hart. Topologically Guaranteed Blobby Surface Polygonization. Proceedings of Western Computer Graphics Symposium. Mar. 1996. pp. 1-9.
- 116.J.C. Hart. On the Hyperbolic Plane and Chinese Checkers. Proceedings of Western Computer Graphics Symposium. Mar. 1996. pp. 69-75
- 117.W.O. Cochran, J.C. Hart, P.J. Flynn. Similarity and Affinity Hashing. Proceedings of Western Computer Graphics Symposium. Mar. 1996. pp. 89-98.
- 118.B.T. Stander, J.C. Hart. Interactive re-polygonization of blobby implicit curves. Proceedings of Western Computer Graphics Symposium. Mar. 1995.
- 119.W.O. Cochran, J.C. Hart, P.J. Flynn. Principal component classification for fractal volume compression. Proceedings of Western Computer Graphics Symposium. Mar. 1995.
- 120.J.C. Hart. Sphere tracing: simple robust antialiased rendering of distance-based implicit surfaces. WSU Technical Report EECS-93-015. Chapter in ``Design, Visualization and Animation of Implicit Surfaces," SIGGRAPH '93 course notes, J. Bloomenthal, B. Wyvill, eds., Aug. 1993.
- 121.J.C. Hart. Ray Tracing Implicit Surfaces. WSU Technical Report EECS-93-014. Chapter in ``Design, Visualization, and Animation of Implicit Surfaces," SIGGRAPH '93 intermediate course notes, J. Bloomenthal and B. Wyvill, eds., Aug. 1993.
- 122.T.A. DeFanti, J.C. Hart, D.J. Sandin, L.H. Kauffman. Visualization of multidimensional fractals. Proceedings of the Argonne National Lab. Seventh International Conference on Advanced Science and Technology. Mar. 1991, pp. 289-298.
- 123.T.A. DeFanti, D.J. Sandin, G.W. Lescinsky, J.C. Hart, M.D. Brown, L.H. Kauffman. Simulacra/Stimulacra::Fractal. Art Futura, Jan. 1991.
- 124.J.C. Hart. Parallel algorithms for visualization of multidimensional fractal objects (extended abstract.) Electronic Imaging East '90. Nov. 1990, pp. 459-462.
- 125.D.J. Sandin, J.C. Hart, L.H. Kauffman. Interactive visualization of complex, stacked and quaternion Julia sets. Proceedings of Ausgraph '90, Sep. 1990. (Invited paper.)
- 126.T.A. DeFanti, D.J. Sandin, J.C. Hart, G.W. Lescinsky. One picture's worth a thousand gigaflops (One animation's worth a thousand teraflops). Pixel 1(3) July 1990, pp. 40-41.
- 127.J.C. Hart, A. Norton. Use of curves in rendering fractures. IBM Technical Report. Aug. 1989.

Film and Animation Credits

- 128.Executive Producer (with C. Machover). "The Story of Computer Graphics." A SIGGRAPH Production of a feature-length documentary on the human struggle to bring computer graphics from the inspired revelation that computers can be used to draw pictures to the present, where low-cost graphics and the Internet combine to provide virtual environments for people to communicate. Premiered at a special event in the Shrine Auditorium, LA, as part of the SIGGRAPH 99 Computer Animation Festival.
- 129.Contributor. "Air on the Dirac Strings." Produced by L.H. Kauffman, G.K. Francis, D.J. Sandin. SIGGRAPH '93 Electronic Theater. SIGGRAPH Video Review 93, Aug. 1993.
- 130.Producer. "SIGGRAPH 93 Technical Session Sampler." SIGGRAPH '93 Electronic Theater. Video compilation, narrated by J. Blinn. SIGGRAPH Video Review 91, Aug. 1993
- 131.Producer. "Highlights from the SIGGRAPH '92 Technical Session." SIGGRAPH '92 Electronic Theater. Video compilation, narrated by J. Blinn.
- 132.Producer/Director/Animator. "Fun with Octrees: Graph Topologies on the Recurrent Cube." SIGGRAPH '92 Animation Screening Room. SIGGRAPH Video Review 87, Aug. 1992. Appears in: "The Fractal Universe: Animation," Media Magic, 1993.
- 133.Producer/Director/Animator. "unNatural Phenomena." SIGGRAPH '91 Electronic Theater. SIGGRAPH Video Review 71, Aug. 1991. Appears in: "The Fractal Universe: Animation," Media Magic, 1993. Excerpts appear in: "Beyond The Minds Eye," Mirimar, and "Prime Time Live," ABC.
- 134.Producer/Director/Animator (with S. Das.). "Sierpinski Blows His Gasket." SIGGRAPH '90 Animation Screening Room. SIGGRAPH Video Review 61, 1990. First Runner-Up, Truevision Videographics Competition, 1990. Appears in: "The Fractal Universe: Animation," Media Magic, 1993.
- 135.Producer/Director/Animator. "Dynamics in the Quaternions." SIGGRAPH Video Review 42, 1989. Appears in: "The Fractal Universe: Animation," Media Magic, 1993.

Educational Activities

Postdoctoral Associates

- Moussa Mansour, 2015-6. Visiting from Univ. Sao Paolo.
- Matei Stroila, 2005-6. Research Scientist, Navteq, Nokia, Here.
- Peer-Timo Bremer, 2004-6. Research Scientist, Lawrence Livermore National Labs.
- Nate Carr, 2004-5. Senior Research Scientist, Adobe.

Ph.D. Students

- Apollo Ellis. Apple Internship 2016.
- Mingcheng Chen (2016). Currently at Google.
- Victor Lu (2013). Currently at Nokia.
- Mahsa Kameya (2012). Currently at Microsoft. MSR Internship 2010.
- Yuntao Jia (2010). Currently at Facebook.
- Jared Hoberock (2008). Research Scientist, Nvidia. NVidia Fellowship 2005,2007. NVidia Internship 2005.

- Xinli Ni (2007). Currently at Google, Microsoft Internship 2005.
- Hui Fang (2006). Currently at Google. Adobe Internship 2005.
- Nate Carr (2004). Senior Research Scientist, Adobe. NVidia Fellowship 2002 and 2003. Intel Internship 1999. NVidia Internship 2002.
- Wayne Cochran (1998). Assistant Professor, School of Engineering and Computer Science, Washington State University, Vancouver.
- Bart Stander (1997), Full Professor, Computer and Information Technology, Dixie State College of Utah.

M.S. Students

- 2023: Sharon Lam. “MWEPI-VIZ: An Interactive Visualization Dashboard for Ensemble Simulations in Computational Epidemiology.”
- 2017: Yiwen Xu.
- 2015: Jared Saul.
- 2013: JohnMark Lau.
- 2011: Rajesh Bhasin.
- 2009: Apeksha Godiyal. Microsoft Visio.
- 2005: Jerry Talton (Ph.D. Stanford, 2012), Tony Kaap (Alias), Matei Stroila (NAVTEQ).
- 2004: Jesse Hall (NVidia, Microsoft Research Internship 2002. NVidia Fellowship 2003 and 2004), Patrick Lacz (RadTIME), Wen Su (NVidia).
- 2002: Hui Fang (continued), Ed Bachta (visualization programmer at IUPUI)
- 2000: Nate Carr (continued), Jay Kim.
- 1999: Chanikarn Kulratanayan.
- 1998: Jeyprakash Michaelraj (Alias).
- 1997: Paul Sherman (Periphery Consulting).
- 1996: Ioana Danciu. (Adobe), Brandon Burch (WSU Instructor), Yingjun Yu (Nokia), Wenbiao Jiang (CFD Research), Kurt Albrecht.
- 1994: Anand Ramagopalrao (Microsoft), Wayne Cochran (continued).
- 1992: Yongqi Yang (Adobe).

Courses Taught

UIUC Courses

- CS 498 Data Visualization (via the Coursera platform for the MCS-DS program)
- ENG 198 Grand Challenges (Fall 2012)
- CS 418 Interactive Graphics
- CS 419 Production Graphics
- CS 519 Scientific Visualization

- CS 497 JCH Advanced Shape Modeling (Fall 2000)
- CS 498 Procedural Modeling and Shading in Computer Graphics (Fall 2002)
- CS 498 Particle System Programming (Spring 2004, voluntary addition to course load)
- CS 498 Shape Modeling (Fall 2004)
- CS 598 JCH Surface Reconstruction (2008)
- CS 491 EH Computational Topology Seminar (co-organized with J. Erickson)
- CS 491 GHY Graphics Seminar (co-organized with M. Garland, Y. Yu)

WSU Courses

- CptS 330 Numerical Computing
- CptS 442 Computer Graphics
- CptS 443 Computer-Human Interaction
- CptS 548 Advanced Computer Graphics

Coursera Courses

- Data Visualization, July-Aug. 2015. Enrollment: 88,399 from 198 different countries, including 1,197 with signature track credit toward the Data Mining certificate.

SIGGRAPH Reviewed One-Day Courses

- 2005: “Modern Techniques for Implicit Modeling” (Terry Yoo, James O’Brien), and “GPU Shading and Rendering” (Marc Olano)
- 2004: “Real Time Shading” (Marc Olano)
- 2003: “Beyond Blobs: Recent Advances in Implicit Surfaces” (Terry Yoo, Greg Turk) and “Real Time Shading” (Marc Olano)
- 2002: “Beyond Blobs: Recent Advances in Implicit Surfaces” (Terry Yoo and Greg Turk, org.) and “Real Time Shading” (Marc Olano).
- 2001: “Real Time Shading” (Marc Olano)
- 2000: “Approaches for Procedural Shading on Graphics Hardware” (Marc Olano)
- 1998: “Procedural Implicit Techniques for Modeling and Texturing” (Co-organizer with D. Ebert)
- 1997: “New Frontiers in Modeling and Texturing” (Co-organizer with D. Ebert.)
- 1996: “Fractal Models for Image Synthesis, Compression and Analysis” (Co-organizer with D. Saupe), “Procedural Modeling, Texturing, and Animation Techniques” (D. Ebert) and “Implicit Surfaces for Geometric Modelling and Computer Graphics (Jai Menon and Brian Wyvill)
- 1995: “Procedural Modeling, Texturing, and Animation Techniques” (D. Ebert).
- 1994: “New Directions for Fractal Modeling in Computer Graphics” (Course organizer).
- 1993: “Design, Visualization and Animation of Implicit Surfaces” (J. Bloomenthal, B. Wyvill).
- 1992: “Fractals: from Folk Art to Hyperreality” (P. Prusinkiewicz).
- 1991: “Fractal Modeling in 3-D Computer Graphics and Imaging” (Co-organizer with K. Musgrave).

Other

- Adler Planerarium, Scientific Visualization by Supercomputer. Chicago, Sept.-Nov. 1988.

Invited Colloquia and Presentations

Note: colloquia presented for the purpose of interviewing have been omitted.

- **Symposium on the Future of Data Analytics of the American & Canadian Societies of Animal Science**, invited talk on “Data Visualization,” July 2018.
- **Workshop High Throughput Phenotyping and Data Analytics**, invited talk on “Data Visualization,” Univ. Maryland, Nov. 2017.
- **31st Discovery Conference on Big Data Dairy Management**, invited talk on “Data Visualization,” Nov. 2016.
- **University of Lorraine**, June 2014.
- **University of Notre Dame**, Center for Research Computing, “High Performance Computer Vision, An Oxymoron?,” May 2012.
- **Graphics Interface Conference**, Keynote: Assistive Technology for the Aesthetically Impaired,” May 2011.
- **University of Iowa Computing Conference**, organized and invited by U. Iowa students, “Assistive Technology for the Aesthetically Impaired,” Feb. 2010.
- **AMS MathViz09**, U. Illinois, “It’s Nice to See This Stuff Actually Used for Something,” Mar. 2009. Host: George Francis.
- **INRIA**, Grenoble, France, “Assistive Technology for the Aesthetically Impaired,” Sep. 2009. Host: Elmar Eisemann.
- **Texas A&M University**. Distinguished Lecture: “Assistive Technology for the Aesthetically Impaired,” Feb. 2008. Host: Scott Schaeffer.
- **NAVTEQ**. “Industry Collaborations,” Nov. 2007, Chicago, IL. Host: Matei Stroila.
- **Adobe-MIT Research Retreat**. “Textureshop Redux,” Oct. 2007, Cambridge, MA. Host: David Salesin and Fredo Durand.
- **University of Konstanz Summerschool**. “Computational Topology of Implicit Surfaces” and “Computational Topology of Meshes,” Sep. 2006, Gaschurn, Austria. Host: Dietmar Saupe.
- **Wolfram Research**. “Mathematical Visualization,” May 2006, Champaign, IL. Host: Roger Germundsson.
- **Carnegie-Mellon University**. “Surface Constrained Particle Systems” and “Global Illumination on the GPU,” Feb. 2005. Host: Doug James.
- **Imagina**. “Global Illumination on the GPU,” Monaco, Feb. 2005.
- **3D Festival**. “Global Illumination on the GPU,” Copenhagen, Denmark, May 2004.
- **INRIA**, Grenoble, France, “Global Illumination on the GPU,” Mar. 2004, Host: Samuel Hornus.
- **University of Utah**. “Global Illumination on the GPU,” Oct. 2003, Host: Chuck Hansen.
- **NVidia U**. Keynote speaker. July 2003, Host: Herbert Hu.

- **Army High Performance Computing Research Center (AHPARC) Workshop on Graphics Modeling, Simulation and Visualization.** “Harnessing the Supercomputer in your PC’s Graphics Card,” June 2003. Host: Boquan Chen.
- **University of Illinois at Chicago (Math Dept.).** “Computational Topology for Computer Graphics.” Apr. 2002. Host: Anatoly Libgober.
- **Colorado Springs SIGGRAPH Chapter.** “The Quest for Real-Time Procedural Solid Texturing.” Mar. 2001. Host: Dino Schweitzer.
- **Northwestern University.** “The Quest for Real-Time Procedural Solid Texturing.” Oct. 2000. Host: Benjamin Watson.
- **Shape Modeling International.** Invited Speaker, Aizu, Japan, March 1999.
- **Universität Freiburg.** “Recurrent Modeling.” Sep. 1997. Host: Dietmar Saupe.
- **University of Utah.** “The Topology of Implicit Surfaces.” Feb. 1997. Host: Chuck Hansen.
- **University of Illinois at Chicago (EECS Dept.).** “The Topology of Implicit Surfaces.” Feb. 1997. Host: Tom DeFanti.
- **Interval Research.** “Recurrent Modeling.” July 1996. Host: Sally Rosenthal.
- **University of Washington.** “Recurrent Modeling.” Dept. of Computer Science and Engineering. Jan. 1996. Host: Linda Shapiro.
- **Boeing Information & Support Services.** “Why Fractal Image Compression Isn’t Fractal.” Computer Graphics & Multimedia Group. Jan. 1996. Host: David Kerlick.
- **University of Idaho.** “Why Fractal Image Compression Isn’t Fractal.” Computer Science Dept. Nov. 1995. Host: Deborah Frincke.
- **NATO Advanced Study Institute** on Fractal Image Compression and Image Analysis. “Similarity Hashing: A model-based vision solution to the inverse problem of recurrent iterated function systems.” July 1995.
- **Simon Fraser University.** “Recurrent Modeling.” School of Computing Science. July 1995. Host: F. David Fracchia.
- **University of Calgary.** “Procedural Geometric Instancing.” Dept. of Computer Science. Jan. 1994. Host: Przemyslaw Prusinkiewicz.
- **Purdue University.** “Linear Fractals in Computer Graphics.” Computer Science Dept. June 1992. Host: Chandrajit Bajaj.

∞ Professional Activities ∞

Editorships and Papers Chairships

- Area Editor (Visual Computing) and Editorial Board Member, ACM Books 2014-2022.
- Papers Co-Chair (with Scott Schaefer), Shape Modeling International, 2012.
- Editor-in-Chief, ACM Transactions on Graphics, 2002-2008. Past EiC, 2008-2011.
- Associate Editor, ACM Transactions on Graphics, 2000-2002, 2011-.
- Guest Editor, Communications of the ACM 35(6) June 1992.

- Guest Editor, SIGGRAPH Video Review 96, ``Video Supplement to the Conference Proceedings," Aug. 1993.
- Guest Editor (with E.E. Catmull), SIGGRAPH Video Review 86, ``Video Supplement to the Conference Proceedings," July 1992.

Professional Organization Leadership

- SIGGRAPH Small Conferences Committee, 2010-
- Chair, SIGGRAPH Service Award, 2005-8.
- ACM SIGGRAPH Advisory Board, 2005.
- Director for Communications, SIGGRAPH Executive Committee, 1996-9.
- Director at Large, SIGGRAPH Executive Committee, 1994-6.

External Research Center Review Panels

- King Abdullah University of Science and Technology, Mar. 2012.
- Science Foundation Ireland, Oct. 2006.
- INRIA Systemes Cognitifs D, May 2006.

Conferences Chaired

- Co-Chair (with A. Marshall-Colon, S.P. Long and E.H. DeLucia), Plants-in-Silico, NCSA, UIUC, 2016.
- Co-Chair (with S. Gao, J. Jorge, K. Polthier and W. Wang), Shape Modeling International, Hong Kong, Oct. 2014.
- Co-Chair (with Marie-Paule Cani), Shape Modeling International, MIT, June 2005.
- Co-Chair (with K. van Overveld), Implicit Surfaces, Eindhoven, Netherlands, 1996.
- Chair. Western Computer Graphics Symposium, Panorama, BC, 1996.

Program Committees

- SIGGRAPH Papers Committee: 2001, 2006, 2007, 2010
- SIGGRAPH Asia Papers Committee, 2008.
- SIGGRAPH Sketches Jury: 1998, 1999.
- Symposium on Geometry Processing, all since 2006.
- Interactive 3D Graphics and Games, all since 2005.
- Pacific Graphics, all since 2002.
- Shape Modeling International, all since 1997.
- Implicit Surfaces (all).
- Eurographics 2004.

- Graphics Hardware, all since 2004.
- Web3D 2003.
- IEEE Virtual Reality 2000, 2001, 2002.
- Virtual Reality Annual International Symposium 1998.
- Visualization and Mathematics 1997.
- Supercomputing 1991-3, 1995.

Memberships

- Association for Computing Machinery
- ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH)
- IEEE Computer Society

University of Illinois Committees

University of Illinois System

- Steering Committee, Start MyResearch Portal for university system-wide research management software, 2014-2016.
- University Technology Management Team, 2012-2013.

Urbana-Champaign Campus

- Ex-Officio (Graduate College Representative), Senate Educational Policy Committee, 2014-.
- Chair, Senate Subcommittee on IT, 2013-6, Member, 2008-2010.
- University Senate Executive Committee, 2013-6.
- IT Governance Executive Committee, 2012-5.
- Chair, IT Governance Research Subcommittee, 2012-5.
- Chair, Stewarding Excellence IT Project Team, 2010.
- Human Factors Transition Committee, 2008-9.
- Senate, 2007-14,

College of Engineering

- Chair, Promotion & Tenure Committee, 2013, Member 2012-2013.
- Vice-Chair, Executive Committee, 2013. Secretary, 2012-2013. Member, 2011-2013.
- Chair, Engineering IT Steering Committee, 2010-11.

Dept. of Computer Science

- Area Chair, Graphics and Human-Computer Interaction, 2004-2011.
- Department Head Search Committee, 2007-9.
- Advisory Committee (peer elected), 2005-2006.
- Industry Relations Committee, 2005-2007.
- Technology, Facilities and Services Committee (CS), 2005-2012.
- Engineering Open House CS Representative (CS), 2001-2006.
- Recruiting Committee, 2004, 2005.
- Curriculum Committee, 2002-3.
- Chair, Distinguished Lecture Series 2001-2002, Member 2000-2002.
- Computer Systems (CS), 2000-2001.
- Fellowships, Assistantships and Admissions, 2000-2001.

WSU

- School of EECS Computer Science Area Chair, 1998-2000.
- School of EECS Computer Science Accreditation Coordinator, 1999-2000.
- School of EECS Curriculum Committee 1998- (Chair, 1999-2000).
- School of EECS Graduate Studies Committee 1993-8.
- School of EECS Laboratory Committee 1994-6.
- WSU Center for Visualization, Analysis and Design in the Molecular Sciences (VADMS) Advisory Board 1994-2000.