



- (51) International Patent Classification:
G06T 15/80 (2011.01)
- (21) International Application Number:
PCT/US2016/016553
- (22) International Filing Date:
4 February 2016 (04.02.2016)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
14/635,280 2 March 2015 (02.03.2015) US
- (71) Applicant: **ADVANCED MICRO DEVICES, INC.**
[US/US]; One AMD Place, Sunnyvale, California 94088-3453 (US).
- (72) Inventors: **OLDCORN, David**; United Business Center, Sentinel House, Ancells Business Park, Harvest Crescent, Fleet GU51 2UZ (GB). **BRENNAN, Chris**; 90 Central Street, Boxborough, Massachusetts 01719 (US). **MAN-TOR, Michael**; 3501 Quadrangle Boulevard, Suite 375, Orlando, Florida 32817 (US). **MAH, Layla A.**; 90 Central Street, Boxborough, Massachusetts 01719 (US).
- (74) Agent: **MATTIOLI, Thomas A.**; Volpe and Koenig, P.C., 30 S. 17th Street, United Plaza, Philadelphia, Pennsylvania 19103 (US).
- (81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,

[Continued on next page]

(54) Title: PROVIDING ASYNCHRONOUS DISPLAY SHADER FUNCTIONALITY ON A SHARED SHADER CORE

(57) Abstract: A method, a non-transitory computer readable medium, and a processor for performing display shading for computer graphics are presented. Frame data is received by a display shader, the frame data including at least a portion of a rendered frame. Parameters for modifying the frame data are received by the display shader. The parameters are applied to the frame data by the display shader to create a modified frame. The modified frame is displayed on a display device.

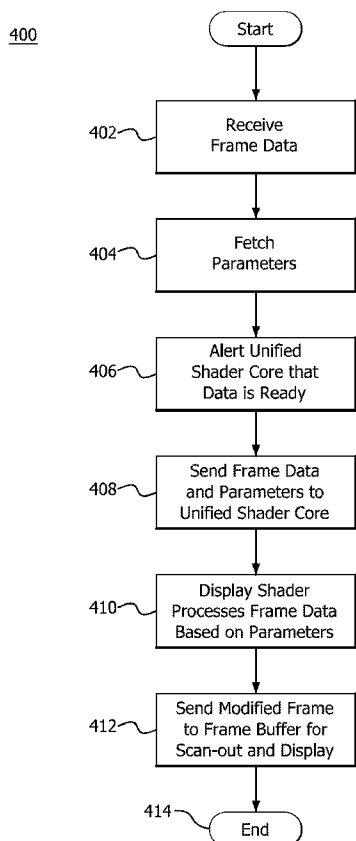


FIG. 4



HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ,

— *with international search report (Art. 21(3))*

PROVIDING ASYNCHRONOUS DISPLAY SHADER FUNCTIONALITY ON A SHARED SHADER CORE

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Application No. 14/635,280 filed March 2, 2015, the contents of which are hereby incorporated by reference herein.

TECHNICAL FIELD

[0002] The disclosed embodiments are generally directed to graphics processing, and in particular, to providing an asynchronous display shader on a shared shader core with multiple input queues.

BACKGROUND

[0003] Currently, when rendering of a 3D frame is completed, the rendered frame is handed off to a display device for display. This process is generally simple — the data is read out from a scan buffer and is sent to the display device.

[0004] Graphics hardware currently includes shader programs that instruct the computer to draw something in a specific way, including applying various effects. A shader may be modified by external parameters provided by the program calling the shader. There are shaders of various types, and each type of shader is applied at a different point in the graphics pipeline. Some shaders are applied when converting the input representations of 3D objects into coordinates of the triangles displayed on-screen that make up a rendered image. Other shaders are applied while each of the individual triangles is being rendered, to map them onto the screen.

[0005] Once a frame is rendered, there is no opportunity to perform additional operations timed to the display refresh(es) after that. This can be emulated with an extra pass after rendering, if the rendering is faster than the display refresh and completes before the display refresh begins. But this cannot be guaranteed given the variable rendering workload.

[0006] This is because rendering occurs at a “rendering rate,” which is variable and based on the 3D rendering workload. Display occurs at a “display

rate,” which happens at the display device’s scan-out rate. A display shader would permit work to be scheduled to be completed at the “display rate” independent of the “rendering rate,” to which there is currently no solution.

[0007] One current solution is to perform the display shading synchronously by waiting until the rendering is complete, running the display shader in one large burst (to quickly let the more rendering begin), and then scheduling the result to be displayed. But this solution requires that all inputs are known when rendering begins and may use one snapshot of the inputs across the entire frame. This may entail waiting for the inputs, which has a long and unpredictable latency and is therefore unacceptable in use cases where low latency is required. To be able to perform computation pacing with the “display rate” that has as little latency from the inputs to scan-out as possible, there needs to be an asynchronous computation that can always access the latest inputs as it performs the scan-out.

[0008] Using a standalone display shader would perform the additional operations closer to real-time by taking the final output of rendering and transforming it on a just-in-time basis before sending it to the display.

SUMMARY OF EMBODIMENTS

[0009] Some embodiments provide a method for performing display shading for computer graphics. Frame data is received by a display shader, the frame data including at least a portion of a rendered frame. Parameters for modifying the frame data are received by the display shader. The parameters are applied to the frame data by the display shader to create a modified frame. The modified frame is displayed on a display device.

[0010] Some embodiments provide a non-transitory computer-readable storage medium storing a set of instructions for execution by a general purpose computer to perform display shading for computer graphics. The set of instructions includes a first receiving code segment, a second receiving code segment, an applying code segment, and a displaying code segment. The first receiving code segment receives frame data by a display shader, the frame data including at least a portion of a rendered frame. The second receiving code

segment receives parameters by the display shader, the parameters for modifying the frame data. The applying code segment applies the parameters to the frame data by the display shader to create a modified frame. The displaying code segment displays the modified frame.

[0011] Some embodiments provide a processor configured to perform display shading for computer graphics. The processor includes a command processor, a shader core, and a shader pipe. The shader core can be shared by multiple processes. The shader pipe is configured to communicate between the command processor and the shader core. A display shader is a program that is sent by the command processor to be executed on the shader core. The display shader is configured to receive frame data, the frame data including at least a portion of a rendered frame; receive parameters for modifying the frame data; and apply the parameters to the frame data to create a modified frame.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] A more detailed understanding may be had from the following description, given by way of example in conjunction with the accompanying drawings, wherein:

[0013] Figure 1 is a block diagram of an example device in which one or more disclosed embodiments may be implemented;

[0014] Figure 2 is a block diagram of an example processor in which one or more disclosed embodiments may be implemented;

[0015] Figure 3 is a flow diagram of data flow to and from a display shader; and

[0016] Figure 4 is a flow chart of a method to process data by the display shader.

DETAILED DESCRIPTION

[0017] A method, a non-transitory computer readable medium, and a processor for performing display shading for computer graphics are presented. Frame data is received by a display shader, the frame data including at least a portion of a rendered frame. Parameters for modifying the frame data are

received by the display shader. The parameters are applied to the frame data by the display shader to create a modified frame. The modified frame is displayed on a display device.

[0018] Figure 1 is a block diagram of an example device 100 in which one or more disclosed embodiments may be implemented. The device 100 may include, for example, a computer, a gaming device, a handheld device, a set-top box, a television, a mobile phone, or a tablet computer. The device 100 includes a processor 102, a memory 104, a storage 106, one or more input devices 108, and one or more output devices 110. The device 100 may also optionally include an input driver 112 and an output driver 114. It is understood that the device 100 may include additional components not shown in Figure 1.

[0019] The processor 102 may include a central processing unit (CPU), a graphics processing unit (GPU), a CPU and GPU located on the same die, or one or more processor cores, wherein each processor core may be a CPU or a GPU. The memory 104 may be located on the same die as the processor 102, or may be located separately from the processor 102. The memory 104 may include a volatile or non-volatile memory, for example, random access memory (RAM), dynamic RAM, or a cache.

[0020] The storage 106 may include a fixed or removable storage, for example, a hard disk drive, a solid state drive, an optical disk, or a flash drive. The input devices 108 may include a keyboard, a keypad, a touch screen, a touch pad, a detector, a microphone, an accelerometer, a gyroscope, a biometric scanner, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals). The output devices 110 may include a display, a speaker, a printer, a haptic feedback device, one or more lights, an antenna, or a network connection (e.g., a wireless local area network card for transmission and/or reception of wireless IEEE 802 signals).

[0021] The input driver 112 communicates with the processor 102 and the input devices 108, and permits the processor 102 to receive input from the input devices 108. The output driver 114 communicates with the processor 102 and the output devices 110, and permits the processor 102 to send output to the output devices 110. It is noted that the input driver 112 and the output driver 114 are

optional components, and that the device 100 will operate in the same manner if the input driver 112 and the output driver 114 are not present.

[0022] Figure 2 is a block diagram of an example processor 200 in which one or more disclosed embodiments may be implemented. It is noted that the processor 200 may include other components not shown in Figure 2; for purposes of discussion, only those portions of the processor relevant to the display shader operation are shown in Figure 2. It is also noted that where there is a plurality of the same element, that element is discussed in the singular to simplify the explanation, but the operation of the element is the same for each of the plurality. To simplify Figure 2, where plural elements communicate with different elements, the communication path is shown via only one of the plural elements.

[0023] The processor 200 includes a plurality of asynchronous compute engine (ACE) command processors (CP) 202₀–202_n. Each ACE CP 202 communicates with a corresponding compute shader (CS) pipe 204₀–204_n. Each CS pipe 204 communicates with a unified shader core 206. The unified shader core 206 communicates with a memory 208. Each ACE CP 202 is capable of adding work into the unified shader core 206 in a prioritized manner.

[0024] A graphics command processor 210 receives and processes graphics commands from an application (not shown in Figure 2). The graphics command processor 210 communicates with the memory 208 and sends work items to a work distributor 212. The work distributor 212 distributes the work items to a CS pipe 214 and to a plurality of primitive pipes 216₀–216_n. Each primitive pipe 216 performs primitive scaling and communicates with the memory 208. Each primitive pipe 216 includes a high order surface shader 218, a tessellator 220, and a geometry shader 222. The high order surface shader 218 provides a high order surface to the tessellator 220, which divides the high order surface into primitives. The primitives are then processed by the geometry shader 222. Both the high order surface shader 218 and the geometry shader 222 communicate with the unified shader core 206.

[0025] The processor 200 also includes a plurality of pixel pipes 224₀–224_n. Each pixel pipe 224 performs pixel scaling and includes a scan converter 226 and

a render backend 228. The geometry shader 222 in the primitive pipe 218 communicates with the scan converter 226 in the pixel pipe 224. The scan converters 226 in each pixel pipe 224 communicate with each other and send data to the unified shader core 206. The render backend 228 communicates with the memory 208 and receives data from the unified shader core 206.

[0026] In the processor 200, the display shader is a shader program executed on the unified shader core 206. The display shader is implemented by duplicating at least a portion of the frame buffer memory (which is part of the memory 208), pointing the display controller at this duplicate frame buffer, and running a just-in-time process in the unified shader core 206 on the data in the original frame buffer to generate the actual output buffer, which is stored in the duplicate frame buffer. In this context, “just-in-time” means that the display shader is run close to real-time after the frame is generated and prior to scan-out and display. The amount of frame buffer memory that needs to be duplicated depends on the display strobe pattern. Duplicating the entire frame buffer memory may not be necessary, but doing so provides a simple implementation.

[0027] The inputs to the display shader are a last generated full 3D frame and the most up-to-date parameters the display shader requires to turn that frame into the display image. It is noted that instead of the last generated full 3D frame, the display shader may receive the last N frames and may also receive depth information, motion information, or more than one layer for composition. The parameters may include, but are not limited to, user interface updates, pointer location, head tracking data, eye tracking data, timestamps for rendered frames, or a current display time. The scope of the parameters supplied to the display shader may be based on an implementation of the display shader selected by a programmer. In one implementation, any information provided to the display shader (including frame data and parameter information) may be provided as pointers to the information, to be retrieved when the display shader is executed on the unified shader core.

[0028] Supplying these inputs to the display shader allows the actual display output to be generated with minimum latency. The frame buffer does not

need to be full before the display shader begins processing the data. A relatively small buffer can be used to begin the process.

[0029] The display shader is executed by loading a program on an ACE CP 202, which submits a high priority request to the unified shader core 206. The submitted work contains the display shading operation. The unified shader core 206 accepts the work from the ACE CP 202 and starts on that work in very short order, due to the high priority request. The display shader must produce its results ahead of the display scan-out. This requires some method to ensure quality of service; examples of quality of service methods are described in greater detail below. It may not be acceptable to wait for other queued work to complete, as the other queued work may require an arbitrary length of time to execute. Once the unified shader core 206 is at least partially free of other work, the priority mechanism in the unified shader core 206 prioritizes the display shader such that it is scheduled ahead of competing workloads.

[0030] One ACE CP 202 may be dedicated to running a display shader initiation process. It tracks the position that the display controller is reading from the post-processed frame buffer, and when it reaches the initiation point, it starts up the display shader process in the unified shader core 206.

[0031] Figure 3 is a flow diagram 300 of data flow to and from a display shader. A frame buffer 302 provides frame data 304 to a display shader 306. The display shader 306 obtains display parameters 308 from memory (not shown in Figure 3) and sends the frame data 304 and the display parameters 308 to a unified shader core 310. The unified shader core 310 executes the display shader 306 to generate a modified frame 312 that is stored in the frame buffer 302. Display data 314 is scanned out of the frame buffer 302 for display on a display device 316.

[0032] In one embodiment, the destination duplicated frame buffer may be limited in size and located on the chip to reduce the power drain of writing data to remote memory and reading the data back in. This embodiment is possible if the results can be guaranteed to be available in time for the scan-out.

[0033] Figure 4 is a flow chart of a method 400 to process data by the display shader. The display shader receives frame data, which is at least a

portion of a 3D frame to be rendered (step 402) and fetches display parameters from memory (step 404). Once the display shader has the necessary frame data and parameters, it alerts the unified shader core that it is ready to execute (step 406).

[0034] Once the ACE CP where the display shader is running receives an indication from the unified shader core that it is available, the ACE CP sends the frame data and the parameters to the unified shader core (step 408), where the display shader processes the frame data based on the parameters (step 410). The processed data is sent from the unified shader core to the frame buffer for scan-out and display on the display device (step 412) and the method terminates (step 414). It is noted that the steps of the method 400 may at least partially overlap. For example, some data could be read from the scan-out buffer for display while other data is concurrently being processed. This means that a portion of a frame can be read out of the buffer and displayed while another portion of the same frame is being processed.

[0035] The display shader makes the latency between the application making the changes and the image appearing on the display as low as possible. This low latency can be achieved because the display shading process takes less time to complete than the original frame rendering. The low latency also enables the display rate to be decoupled from the rendering rate. The display shader may be run at high priority, to guarantee minimum latency or low priority, to minimize the impact on other workloads. If run at low priority, the initiation point must be adjusted earlier to ensure that the shader has time to complete.

[0036] Because the display shader is implemented as a just-in-time process, there needs to be some sort of quality of service (QoS) guarantee. If the display shader does not complete its processing on time, then the display scan runs ahead of the data in the scan-out buffer and “garbage” (i.e., the wrong data) is displayed on the screen.

[0037] There needs to be a high level of confidence that the display shader can complete its work in the latency time allowed. There is no hard limit as to the permitted latency time allowed, but the display shader needs to complete its work close to the predicted length of time, very nearly all the time. The

prioritization in the unified shader core helps to meet the QoS guarantee. With the prioritization, the display shader can effectively “take over” the entire shader core until it has completed its operations.

[0038] In one implementation, the unified shader core will wait until any existing work is completed before executing the display shader, even though the display shader is run with a high priority. In a second implementation, work currently underway in the unified shader core may be interrupted, to permit the display shader to run. In a third implementation, there may be room on the unified shader core to run the display shader, even if there is existing work currently being done.

[0039] In a fourth implementation, resources for the display shader can be pre-reserved, such that when the display shader is ready to run, it can run and does not need to wait for existing work on the unified shader core to complete. In this implementation, the work is not scheduled onto the ACE CP until it is known that the data is ready. Alternatively, if the data is transient, the data may be updated in a dynamic manner during the display shading process.

[0040] There are several possible ways to keep the initiator process running on the ACE CP:

[0041] (1) Use an existing streaming engine and regularly feed it with new instances of the initiator process, each of which sleeps until the initiation point and terminates afterwards. If the operating system (OS) provides queues which automatically fill the ACE CP when a previous process retires, this can be achieved by using a CPU connected to the graphics controller, as long as the worst-case latency of process start is longer than the interval between initiations and the cost of rescheduling is not too high.

[0042] (2) Start and stop a looped continuous process on the ACE CP. This method might not be acceptable if GPU processes are required to exit in a finite amount of time, which is true on some OSes.

[0043] (3) A hybrid of the above: the ACE CP process is scheduled once per frame, and the single process loops and executes a fixed number of initiation points before exiting.

[0044] The pattern of display shader execution needs to be matched to the pattern of strobe on the display device if minimum latency is to be maintained. For example, if the display is strobed in one pass, the display shader needs to execute once per display frame. If the display is strobed top half and bottom half, the display shader executes twice per frame, once on each half. If the display is continuously strobed, the display shader ideally would be executed per pixel, but in realistic circumstances is likely to be executed every few display scan lines. To determine the strobe pattern of the display device, the display shader may communicate with the device or the pattern may be set by a programmed table or assumption.

[0045] For most display shading algorithms, there is a method of snapshotting the input parameters at initiation time. Not all parameters may need to be updated precisely simultaneously; generally, groups of parameters will require an atomic update (e.g., a transformation matrix or the buffer location of a previously completed frame to be processed).

[0046] In a system with multiple GPUs (including an accelerated processing unit (APU) and GPU combination), the display shader need only execute on one GPU. It is usually most convenient to execute the display shader on the GPU which is closest to the display port or display controller, to avoid the latency cost of transfer across a slow system bus, although this is not a requirement.

[0047] The display shader may be used in various situations, including, but not limited to:

[0048] (1) Asynchronous time warping for virtual reality headset display latency reduction.

[0049] (2) Other low latency composition, including mouse pointer overlays of higher complexity or frame rate conversion. For example, with a 4K display device, there may be a large and complicated cursor. During game play, a player desires an instantaneous cursor response; any latency in moving the cursor around the screen would adversely affect game play.

[0050] (3) Temporal antialiasing and frame accumulation.

[0051] (4) Motion compensated frame rate conversion.

[0052] It should be understood that many variations are possible based on the disclosure herein. Although features and elements are described above in particular combinations, each feature or element may be used alone without the other features and elements or in various combinations with or without other features and elements.

[0053] The methods provided may be implemented in a general purpose computer, a processor, a processor core, or the display device. Suitable processors include, by way of example, a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs) circuits, any other type of integrated circuit (IC), and/or a state machine. Such processors may be manufactured by configuring a manufacturing process using the results of processed hardware description language (HDL) instructions and other intermediary data including netlists (such instructions capable of being stored on a computer readable media). The results of such processing may be maskworks that are then used in a semiconductor manufacturing process to manufacture a processor which implements aspects of the embodiments.

[0054] The methods or flow charts provided herein may be implemented in a computer program, software, or firmware incorporated in a non-transitory computer-readable storage medium for execution by a general purpose computer or a processor. Examples of non-transitory computer-readable storage mediums include a read only memory (ROM), a random access memory (RAM), a register, cache memory, semiconductor memory devices, magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks, and digital versatile disks (DVDs).

* * *

CLAIMS

What is claimed is:

1. A method for performing display shading for computer graphics, comprising:
 - receiving frame data by a display shader, wherein the frame data includes at least a portion of a rendered frame;
 - receiving parameters by the display shader, the parameters for modifying the frame data;
 - applying the parameters to the frame data by the display shader to create a modified frame; and
 - displaying the modified frame.
2. The method according to claim 1, wherein the display shader is executed on a shader core which can be shared by multiple processes.
3. The method according to claim 2, wherein the shader core includes a priority mechanism wherein the display shader can be executed with a higher priority than other processes on the shader core.
4. The method according to claim 2, further comprising:
 - alerting the shader core that the display shader is ready to execute.
5. The method according to claim 1, wherein the displaying includes:
 - storing the modified frame in a buffer by the display shader; and
 - reading the modified frame from the buffer to be displayed.
6. A non-transitory computer-readable storage medium storing a set of instructions for execution by a general purpose computer to perform display shading for computer graphics, the set of instructions comprising:

a first receiving code segment for receiving frame data by a display shader, wherein the frame data includes at least a portion of a rendered frame;

a second receiving code segment for receiving parameters by the display shader, the parameters for modifying the frame data;

an applying code segment for applying the parameters to the frame data by the display shader to create a modified frame; and

a displaying code segment for displaying the modified frame.

7. The non-transitory computer-readable storage medium according to claim 6, further comprising:

an alerting code segment for alerting a shader core that the display shader is ready to execute.

8. The non-transitory computer-readable storage medium according to claim 6, wherein the displaying code segment includes:

a storing code segment for storing the modified frame in a buffer by the display shader; and

a reading code segment for reading the modified frame from the buffer to be displayed.

9. The non-transitory computer-readable storage medium according to claim 6, wherein the instructions are hardware description language (HDL) instructions used for the manufacture of a device.

10. A processor configured to perform display shading for computer graphics, comprising:

a command processor;

a shader core which can be shared by multiple processes; and

a shader pipe, configured to communicate between the command processor and the shader core,

wherein a display shader is a program that is sent by the command processor to be executed on the shader core, the display shader configured to:

receive frame data, wherein the frame data includes at least a portion of a rendered frame;

receive parameters for modifying the frame data; and

apply the parameters to the frame data to create a modified frame.

11. The processor according to claim 10, wherein the shader core includes a priority mechanism wherein the display shader can be executed with a higher priority than other processes on the shader core.

12. The processor according to claim 10, wherein the command processor is configured to alert the shader core that the display shader is ready to execute.

13. The processor according to claim 10, further comprising:
a buffer, configured to receive the modified frame from the display shader.

14. A non-transitory computer-readable storage medium storing a set of instructions for execution by one or more processors to facilitate manufacture of a processor configured to perform display shading for computer graphics, the processor comprising:

a command processor;

a shader core which can be shared by multiple processes; and

a shader pipe, configured to communicate between the command processor and the shader core,

wherein a display shader is a program that is sent by the command processor to be executed on the shader core, the display shader configured to:

receive frame data, wherein the frame data includes at least a portion of a rendered frame;

receive parameters for modifying the frame data; and

applying the parameters to the frame data to create a modified frame.

15. The non-transitory computer-readable storage medium according to claim 14, wherein the shader core includes a priority mechanism wherein the display shader can be executed with a higher priority than other processes on the shader core.

16. The non-transitory computer-readable storage medium according to claim 14, the processor further comprising:

a buffer, configured to receive the modified frame from the display shader.

17. The non-transitory computer-readable storage medium according to claim 14, wherein the instructions are hardware description language (HDL) instructions used for the manufacture of a device.

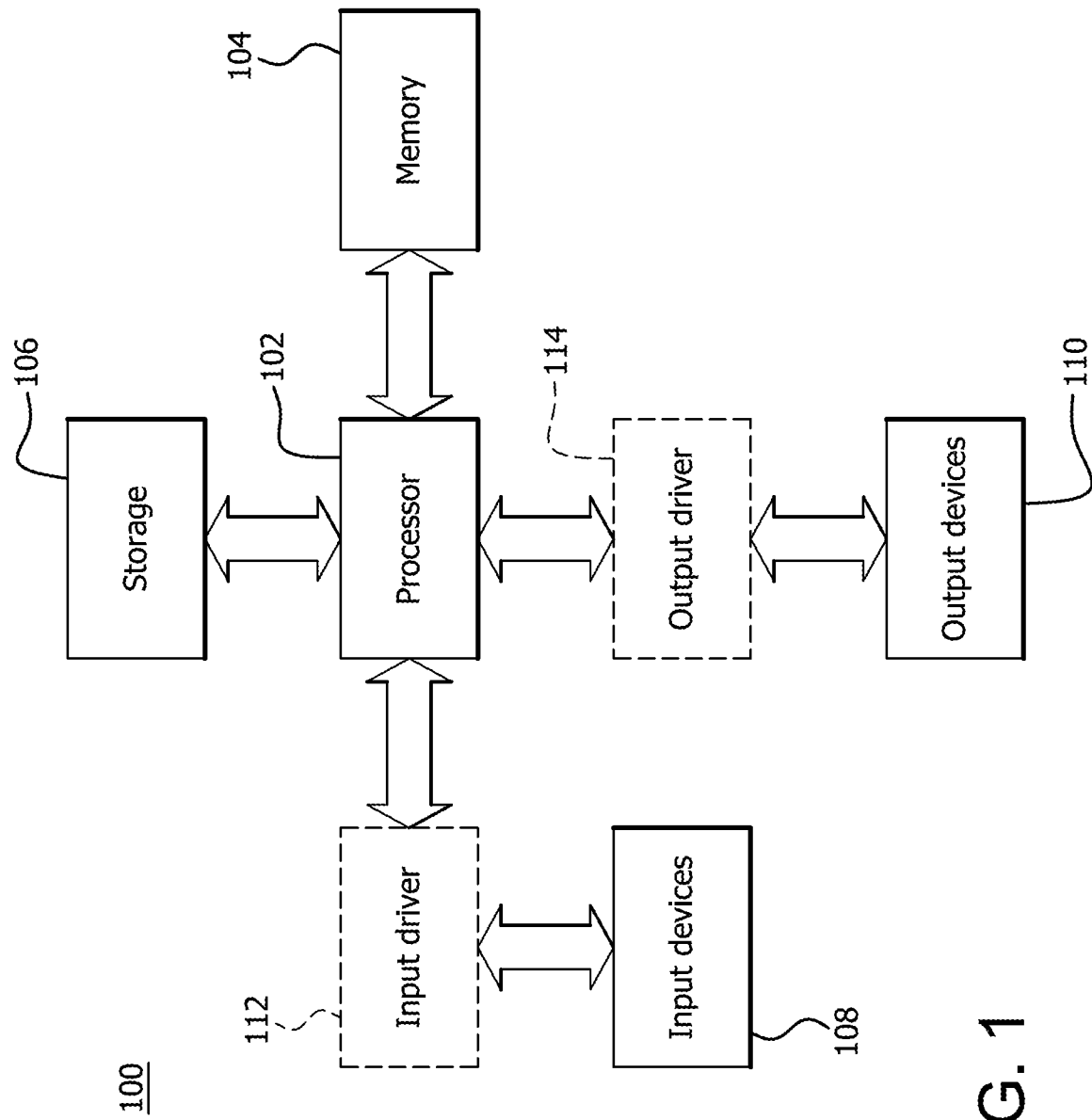


FIG. 1

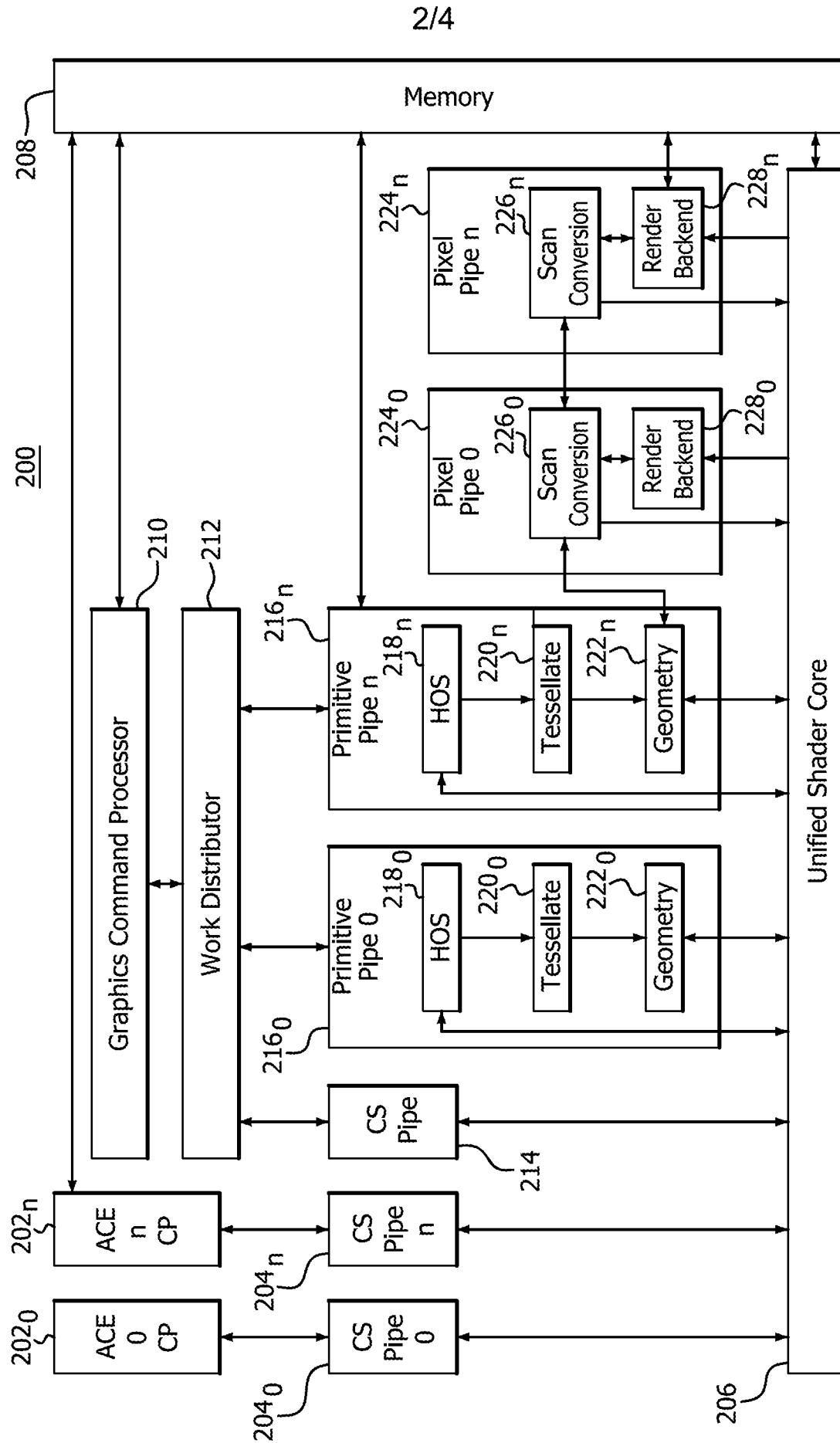


FIG. 2

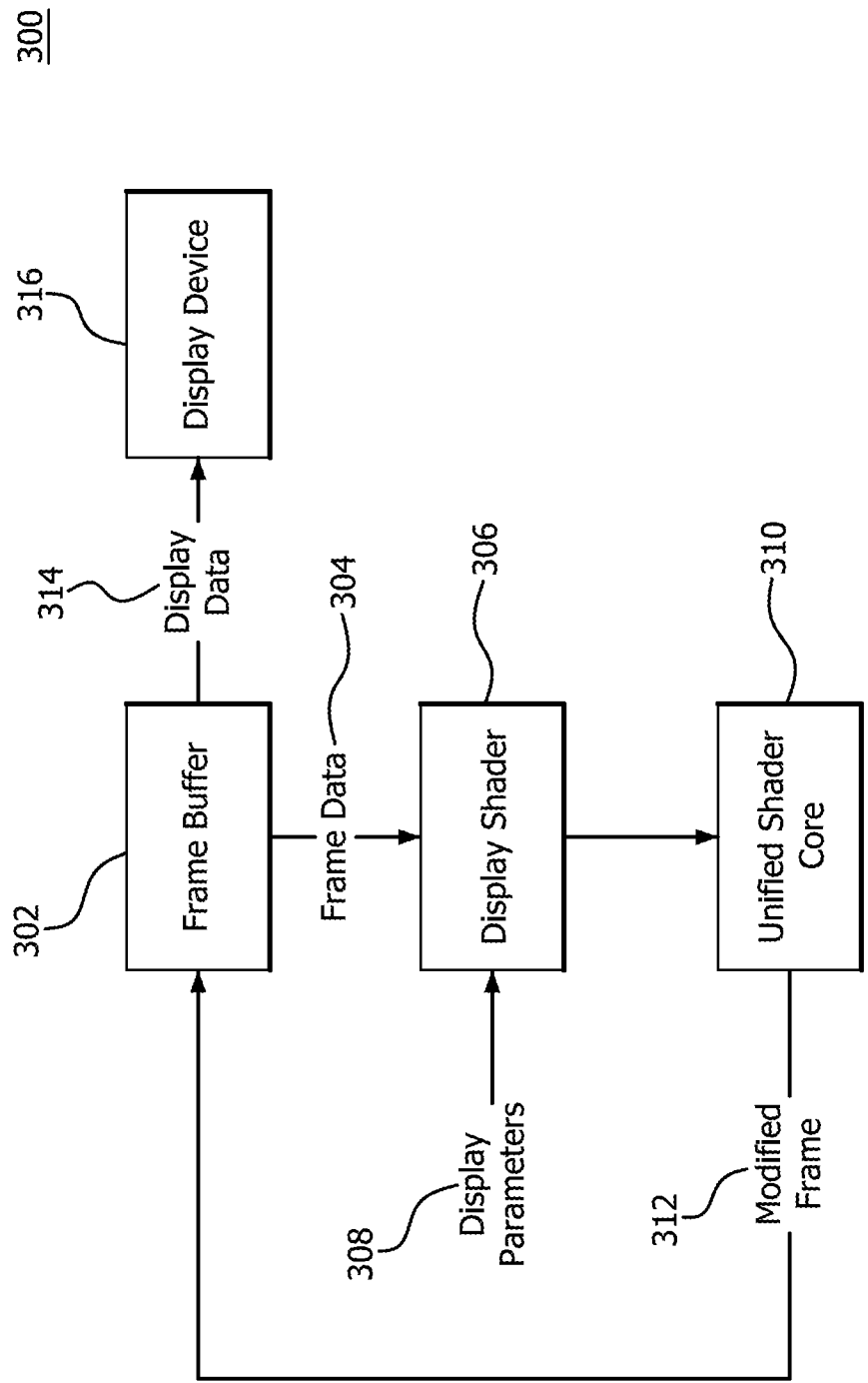


FIG. 3

4/4

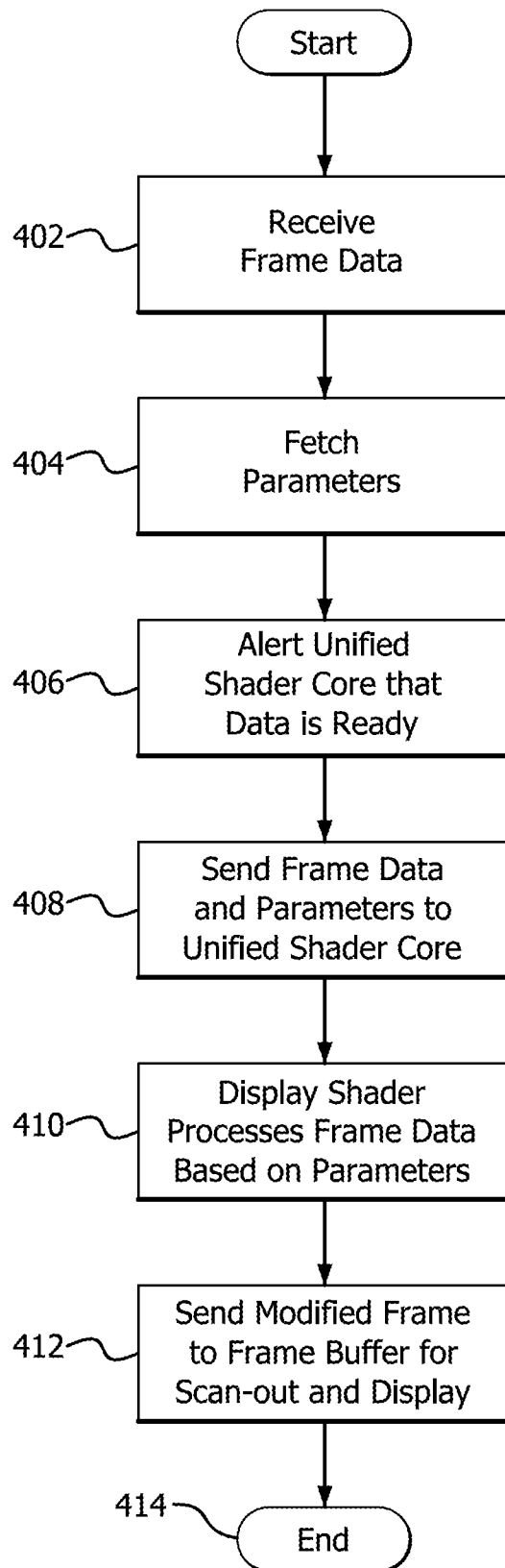
400

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/016553**A. CLASSIFICATION OF SUBJECT MATTER****G06T 15/80(2011.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06T 15/80; G06T 15/60; G06F 9/06; G06T 1/20; G06F 15/76; G06T 1/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: graphic processor, frame buffer, shader core, parameter, modified data, output image

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2012-0017062 A1 (VINEET GOEL et al.) 19 January 2012 See paragraphs [0005], [0026]-[0028], [0040]-[0041], [0060]; and figure 1.	1-17
Y	US 2009-0219288 A1 (ALAN BRYANT HEIRICH) 03 September 2009 See paragraphs [0027]-[0028], [0035]-[0045]; claim 1; and figures 1, 3.	1-17
Y	US 2013-0155077 A1 (ROBERT SCOTT HARTOG et al.) 20 June 2013 See paragraphs [0039], [0080]; claim 1; and figures 1A-1B.	3, 11, 15
A	US 2010-0214301 A1 (JINYU LI et al.) 26 August 2010 See paragraphs [0044]-[0048]; claim 21; and figure 4.	1-17
A	WO 2013-133957 A1 (QUALCOMM INCORPORATED) 12 September 2013 See paragraphs [0024]-[0065], [0071]-[0076]; claims 1-3; and figures 1-2, 4.	1-17



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

03 June 2016 (03.06.2016)

Date of mailing of the international search report

03 June 2016 (03.06.2016)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

LEE, Dong Yun

Telephone No. +82-42-481-8734



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/016553

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2012-0017062 A1	19/01/2012	CN 103003838 A EP 2596470 A1 JP 2013-541748 A KR 10-2013-0141446 A WO 2012-012440 A1	27/03/2013 29/05/2013 14/11/2013 26/12/2013 26/01/2012
US 2009-0219288 A1	03/09/2009	US 2006-0152509 A1 US 7548244 B2 US 7907143 B2	13/07/2006 16/06/2009 15/03/2011
US 2013-0155077 A1	20/06/2013	CN 103999051 A EP 2791795 A2 JP 2015-502618 A KR 10-2014-0101384 A WO 2013-090773 A2 WO 2013-090773 A3	20/08/2014 22/10/2014 22/01/2015 19/08/2014 20/06/2013 08/08/2013
US 2010-0214301 A1	26/08/2010	US 8711159 B2	29/04/2014
WO 2013-133957 A1	12/09/2013	CN 104160420 A EP 2823459 A1 EP 2823459 B1 JP 2015-515052 A JP 5792402 B2 KR 10-1552079 B1 KR 10-2014-0138842 A US 2013-0235053 A1 US 9019289 B2	19/11/2014 14/01/2015 17/02/2016 21/05/2015 14/10/2015 09/09/2015 04/12/2014 12/09/2013 28/04/2015