(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 June 2013 (20.06.2013)

W I P O I P C T

(10) International Publication Number
WO 2013/090435 A1

(51) **International Patent Classification:**
G06T 1/20 (2006.01)

(21) **International Application Number:**
PCT/US20 12/069249

(22) **International Filing Date:**
12 December 2012 (12. 12.2012)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
13/324,437 13 December 201 1 (13. 12.201 1) US

(71) **Applicant: ADVANCED MICRO DEVICES, INC.** [US/US]; One AMD Place, Sunnyvale, CA 94088 (US).

(72) **Inventor: ROGERS, Philip, J.;** 158 Heald Street, Pepperell, MA 01463 (US).

(74) **Agents: SPECHT, Michael, D.** et al; Sterne, Kessler, Goldstein & Fox PLLC, 1100 New York Avenue, N.W., Washington, DC 20005-3934 (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, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, 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.

(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, SZ, 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, ML, MR, NE, SN, TD, TG).

[Continued on nextpage]

(54) **Title:** SOFTWARE CONSTANTS FILE

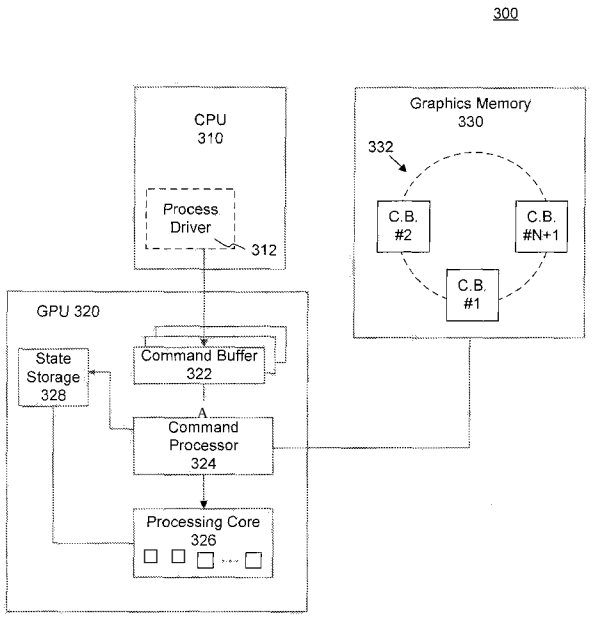


FIG. 3

(57) **Abstract:** Methods and systems relating to providing constants are provided. In an embodiment, a method, of providing constants in a processing device includes copying a constant of a. first constant buffer to a second constant buffer, the first and second constant buffers being included, in a ring of constant buffers and a size of the ring being one greater than a number of processes that the processing device can process concurrently, updating a value of the constant in the second buffer, and binding a command to be executed on the processing device to the second constant buffer.

WO 2013/090435 A1

Published:

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

SOFTWARE CONSTANTS FILE

BACKGROUND

Field

[0001] The present invention relates to providing constants in processing systems.

Background Art

[0002] Many processing systems include a central processing unit (CPU) and a graphics processing unit (GPU). The CPU is a largely serial device that processes instructions sequentially, with subsequent instructions often relying on the results of previous instructions. The GPU, on the other hand, is a highly parallelized device, often including multiple execution units. Through its parallelized architecture, a GPU can be especially apt at executing graphics commands that often include the same operation executed on a multitude of different data sets (e.g., operations to change the state of pixels of a display).

[0003] Early GPUs included a hardware storage that stored a single set of state associated with commands being executed by the GPU. This state information can include, for example, constants that the command will access during execution. Thus, every time a command associated with different state information had to be executed, the launching of that command would have to wait for the previous command to finish executing so that the state information could be safely overwritten without affecting the command currently in flight.

[0004] To overcome the limitations of these early GPUs, GPUs were developed that include hardware storage able to store a number of different sets of state information (e.g., GPUs that implemented DirectX 9). For example, some GPUs have hardware storage able to store eight sets of state information. Each of these sets of information included a number of constants that the command will access during execution. This hardware storage, however, can be expensive in terms of board or die space. Thus, many newer devices have eliminated hardware-based constants storage (e.g., GPUs that implement DirectX 10). Newer programs, associated with these newer devices, have been written to allocate portions of memory the execution engine can access when constants are needed. Older programs configured to access stored hardware-based constants, however, cannot be executed on these newer devices.

BRIEF SUMMARY OF EMBODIMENTS

[0005] Methods and systems relating to providing constants are provided. In an embodiment, a method of providing constants in a processing device includes copying a constant of a first constant buffer to a second constant buffer, the first and second constant buffers being included in a ring of constant buffers and a size of the ring being one greater than a number of processes that the processing device can process concurrently, updating a value of the constant in the second buffer, and binding a command to be executed on the processing device to the second constant buffer.

[0006] In another embodiment, a processing system includes a command processor configured to process commands for N processes simultaneously and a memory configured to store N+1 constant buffers. The command processor is configured to copy a constant of a first constant buffer of the N+1 buffers to a second constant buffer of the N+1 constant buffers, update a value of the constant in the second buffer, and bind a received command to the second constant buffer.

[0007] Further features and advantages of the invention, as well as the structure and operation of various embodiments of the invention, are described in detail below with reference to the accompanying drawings. It is noted that the invention is not limited to the specific embodiments described herein. Such embodiments are presented herein for illustrative purposes only. Additional embodiments will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein.

BRIEF DESCRIPTION OF THE DRAWINGS/FIGURES

[0008] The accompanying drawings, which are incorporated herein and form part of the specification, illustrate the present invention and, together with the description, further serve to explain the principles of the invention and to enable a person skilled in the pertinent art to make and use the invention. Various embodiments of the present invention are described below with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout.

[0009] FIG. 1 is an illustrative block diagram of a conventional processing system.

[0010] FIG. 2 is an illustrative block diagram of a conventional state storage.

[0011] FIG. 3 is an illustrative block diagram of a processing system, according to an embodiment of the present invention.

[0012] FIG. 4 is a flowchart illustrating a method for providing constants, according to an embodiment of the present invention.

[0013] FIG. 5 is an illustrative block diagram of a ring buffer, according to an embodiment of the present invention.

DETAILED DESCRIPTION

[0014] It is to be appreciated that the Detailed Description section, and not the Summary and Abstract sections, is intended to be used to interpret the claims. The Summary and Abstract sections may set forth one or more but not all exemplary embodiments of the present invention as contemplated by the inventor(s), and thus, are not intended to limit the present invention and the appended claims in any way.

[0015] FIG. 1 is a block diagram illustration of a conventional processing system 100. Processing system 100 includes a CPU 110 and a GPU 120. GPU 120 includes command buffers 122, a command processor 124, a processing core 126, and a state storage 128.

[0016] CPU 110 can have a number of applications running thereon. One or more of these applications can generate and send commands (e.g., draw commands) to GPU 120 for execution. Draw commands often include the same operation executed on a number of different sets of data. For example, draw commands can include changes to the state of the pixels of a display. Because GPU 120 is a highly parallelized device, GPU 120 can be especially well-suited to handle these types of commands.

[0017] To facilitate interaction between CPU 110 and GPU 120, CPU 110 includes a number of process drivers 112. In an embodiment, a process driver is provided for each process running on CPU 110 that sends commands to GPU 120. FIG. 1 shows a single process driver 112 for simplicity. However, one skilled in the art would appreciate that multiple process drivers, each for a different application running on CPU 110. Based on the needs of its corresponding application, process driver 112 can create command buffers 122. For example, each of the command buffers can include a number of commands that specify the relevant state values (e.g., the values of one or more constants, depth, a texture, etc.) and command(s) to be executed on the processing core using that state information (e.g., a draw command).

[0018] Command processor 124 retrieves commands from command buffers 122. For the commands relating to setting state information, command processor sets the relevant values in state storage 128. FIG. 2 shows a schematic diagram of conventional state storage 128 implemented, for example, according to DirectX 9. As shown in FIG. 2, state storage 128 includes N contexts 202a-n (collectively "contexts 202"). Contexts 202 include a respective one of constants 204a-n (collectively "constants 204"). When command processor 124 retrieves a command from command buffer 122, it can update portions of one of the contexts 202 that is not currently being used by a command currently in flight. Thereafter, when that command is put into flight, it can operate using the state information stored in its respective context 202. In particular, constants 204 remain constant in a command's respective context throughout the operation of the command on processor core 126.

[0019] As far as command(s) of the command buffer that are to be executed by processing core 126, command processor 124 delivers those commands to processing core 126, where they will be executed using a respective one of contexts 202. In an embodiment, command processor 124 can be a microprocessor or microcontroller. Command processor 124 configurable for selecting command buffers 122 based on inputs received from a run list controller (RLC) (not shown). The RLC can be controlled by an operating system running on CPU 110.

[0020] Processing core 126 is a highly parallelized processing device that is especially suited to execute drawing commands in which the same operation is conducted on a number of different sets of data. For example, as shown in FIG. 1, processing core 126 can include a multitude of individual execution modules 127.

[0021] As noted above, state storage 128 (including constants 204) is implemented in hardware. Maintaining constant values in hardware, however, is expensive in terms of silicon board space. Thus, many systems have eliminated the use of state storage 128 to store constants (e.g., systems that implement DirectX 10). Instead, newer applications running on CPU 110 are configured to use memory (not shown in FIG. 1) to store constants needed during execution. Although these systems may overcome the drawbacks of hardware-based constant storage, they may be unable to efficiently execute certain programs configured to use these hardware-based constants.

[0022] In embodiments described herein, a ring of constant buffers is provided for storing constants associated with commands running on a processing core. As explained in greater detail below, the ring of constant buffers is sized so that when updates are made, none of the constant buffers currently in use by commands running on the processing core are affected. In this manner, even newer processing systems that eliminate expensive hardware-based constant storage can maintain backwards compatibility with older legacy programs that still access hardware-based stored constants.

[0023] FIG. 3 is a block diagram illustration of a processing system 300, according to an embodiment of the present invention. The processing system 300 includes a CPU 310, a GPU 320, and a graphics memory 330. CPU 310 includes a process driver 312 running thereon. Process driver 312 corresponds to an application running on CPU 310. GPU 320 includes command buffers 322, a command processor 324, a processing core 326, and a state storage 328.

[0024] Graphics memory 330 includes a ring of constant buffers 332. Ring of constant buffers 332 can be initialized by command processor 324. Ring of constant buffers 232 includes N+l constant buffers, where the value N represents the maximum possible number of contexts that state storage 328 can store simultaneously. In an embodiment, N is eight. As would be apparent to those skilled in the relevant arts based on the description herein, the value of N can take on any number of different values without departing from the scope and spirit of the present invention. Although FIG. 3 shows CPU 310, GPU 320, and graphics memory 330 implemented separately, those skilled in the art will recognize that these components can be integrated together on a single die. The operation of system 300 will be described in greater detail with reference to FIG. 4.

[0025] FIG. 4 is a flowchart 400 of an exemplary method of practicing an embodiment of the present invention. More specifically, the flowchart 400 includes example steps for providing constant values, according to an embodiment of the present invention. Other structural and operational embodiments will be apparent to persons skilled in the relevant art(s) based on the following discussion. The steps shown in FIG. 4 are not necessarily required to occur in the order shown. The steps of FIG. 4 are described in detail below.

[0026] In step 402, a ring of constant buffers is created. For example, in FIG. 3 command processor 324 can create ring 332 in graphics memory 330. As shown in FIG. 3, ring 332 includes N + l constant buffers, where N is the number of contexts that state

storage 328 can store simultaneously. Including at least N + 1 constant buffers in ring 332 ensures that a constant buffer can be overwritten with new values without affecting any commands that are in the graphics pipeline because there can be, at most, N different commands having different state being executed on GPU 320 at a single time.

[0027] In step 404, a command buffer is received. For example, in FIG. 3, process driver 312 can generate one or more of command buffers 322 in response to input from its corresponding application running on CPU 310. For example, the application can be game running on CPU 310 and the command buffer can include a command to draw an image in response to an input from a user.

[0028] In step 406, at least one constant of a first constant buffer is copied to a second constant buffer. For example, in FIG. 3, command processor 324 can select one of command buffers 322. In addition to including commands to be executed by processing core 326, the command buffer can also include commands that specify the state to be used in executing those commands. However, in contrast to the command buffers includes in conventional system shown in FIG. 2, command buffers 322 include a command to copy the contents of the most recently updated constant buffer to a constant buffer not currently in use (e.g., the constant buffer located N buffers away in the clockwise direction in FIG. 3).

[0029] In an embodiment, command processor 324 can copy all the constants included in the first constant buffer to the second constant buffer. Because the drawing command may rely on constants that have not been updated as well as constants that have been updated, all constants that will be needed for operation are ensured of being included in the second constant buffer.

[0030] In another embodiment, command processor 324 can be configured to establish a "high water mark" prior to executing the copy. For example, command process 324 can track the number of constants that a process uses. If command processor 324 determines that the process does not use constants beyond a certain point in a command buffer, termed a "high water mark," command processor 324 may only copy constants up to that high water mark.

[0031] For example, FIG. 5 is an exemplary illustration of a ring of constant buffers 500, according to an embodiment of the present invention. In accordance with the embodiment of FIG. 5, command processor 324 determines that the application only uses

constants between constant 0 and constant y, leaving the others undefined. Thus, when executing the copy, instead of copying all the values of constant buffer 1 to constant buffer N + 1, command processor 324 can instead only copy constants between constant 0 and constant y, thereby saving processing time.

[0032] In another embodiment, command processor 324 can establish a "dynamic high water mark." In establishing the dynamic high water mark, command processor 324 can identify values for the same constant for all constant buffers within the ring. If all the constant buffers have the same value for a particular constant, command processor 324 can omit copying that constant. For example, in FIG. 5, command processor 324 can omit copying constant 0 if the value of constant 0 is the same for all of the constant buffers in ring 500. Like the high water mark described above, the establishing of a dynamic high water mark reduces the number of values that must be copied, thereby reducing processing time and enhancing overall performance.

[0033] In step 408, the second constant buffer is updated. For example, in FIG. 3, command processor 324 can update the second constant buffer based a command included in the command buffer. For example, the command buffer can include a command to update a particular value of the second constant buffer. For example, the application to which process driver 312 corresponds can generate a constant to be used by the command when executing and command processor can overwrite a constant in the second constant buffer to that value.

[0034] In step 410, the command is bound to the second constant buffer. For example, in FIG. 2, command processor 324, responsive to another command in the command buffer, can bind the held command to be executed on processing core 326 to the second command buffer.

[0035] In step 412, the command is executed. For example, in FIG. 3, the processing core 226 can execute the draw command. In executing the draw command, the command, which has been binded to command buffer N+l, processing core 226 can access command buffer N+l when constants are needed during execution of the command.

[0036] The present invention has been described above with the aid of functional building blocks illustrating the implementation of specified functions and relationships thereof. The boundaries of these functional building blocks have been arbitrarily defined herein

for the convenience of the description. Alternate boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed.

[0037] The foregoing description of the specific embodiments will so fully reveal the general nature of the invention that others can, by applying knowledge within the skill of the art, readily modify and/or adapt for various applications such specific embodiments, without undue experimentation, without departing from the general concept of the present invention. Therefore, such adaptations and modifications are intended to be within the meaning and range of equivalents of the disclosed embodiments, based on the teaching and guidance presented herein. It is to be understood that the phraseology or terminology herein is for the purpose of description and not of limitation, such that the terminology or phraseology of the present specification is to be interpreted by the skilled artisan in light of the teachings and guidance.

[0038] The breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

WHAT IS CLAIMED IS:

1. A method of providing constants in a processing device, comprising:

creating a ring of constant buffers in a memory coupled to the processing device;

responsive to a received command:

copying a constant of a first constant buffer of the ring to a second constant buffer of the ring;

updating a value of the constant in the second buffer; and

binding a command to be executed on the processing device to the second constant buffer.

2. The method of claim 1, further comprising:

executing the command using the processing device.

3. The method of claim 2, wherein executing comprises:

accessing the constant from the second constant buffer.

4. The method of claim 1, wherein the ring of constant buffers is stored in a graphics memory.

5. The method of claim 1, further comprising:

identifying a maximum level of a constant buffer that an application has previously defined.

6. The method of claim 5, wherein copying comprises:

copying a plurality of constants of the first constant buffer to the second constant buffer, wherein the constant is included in the plurality of constants and wherein the number of constants in the plurality of constants is equal to the maximum level.

7. The method of claim 1, wherein copying comprises:

copying only a plurality of constants of the first constant to the second constant buffer, wherein the constant is included in the plurality of constants and wherein constants of the plurality of constants have values different from respective constants of the second constant buffer.

8. The method of claim 1, further comprising:

holding the command in a command buffer.

9. The method of claim 1, wherein the copying step is executed responsive to a second command.

10. The method of claim 1, wherein the number of constant buffers are at least one greater than the number of contexts available on the processing device

11. A processing system, comprising:

a memory configured to store a ring;

a command processor is configured to, responsive to a received command buffer:

copy a constant of a first constant buffer of the ring to a second constant buffer of the ring;

update a value of the constant in the second buffer; and

bind a command to be executed on the processing device to the second constant buffer.

12. The processing system of claim 11, further comprising:

a processing core configured to execute the command.

13. The processing system of claim 12, wherein the processing core is configured to access the constant from the second constant buffer.

14. The processing system of claim 11, wherein the command processor is configured to identify a maximum level of a constant buffer that an application has previously defined.

15. The processing system of claim 14, wherein the command processor is configured to copy a plurality of constants of the first constant buffer to the second constant buffer, wherein the constant is included in the plurality of constants and wherein the number of constants in the plurality of constants is equal to the maximum level.

16. The processing system of claim 11, wherein the command processor is configured to copy only a plurality of constants of the first constant to the second constant buffer, wherein the constant is included in the plurality of constants and wherein constants of the plurality of constants have values different from respective constants of the second constant buffer.

17. The processing system of claim 11, further comprising:

a command buffer configured to hold the command in a buffer.

18. The processing system of claim 11, wherein command processor is configured to copy the constant responsive to a second command received from a central processing unit.

19. The processing system of claim 11, wherein the ring has N+1 constant buffers.

100

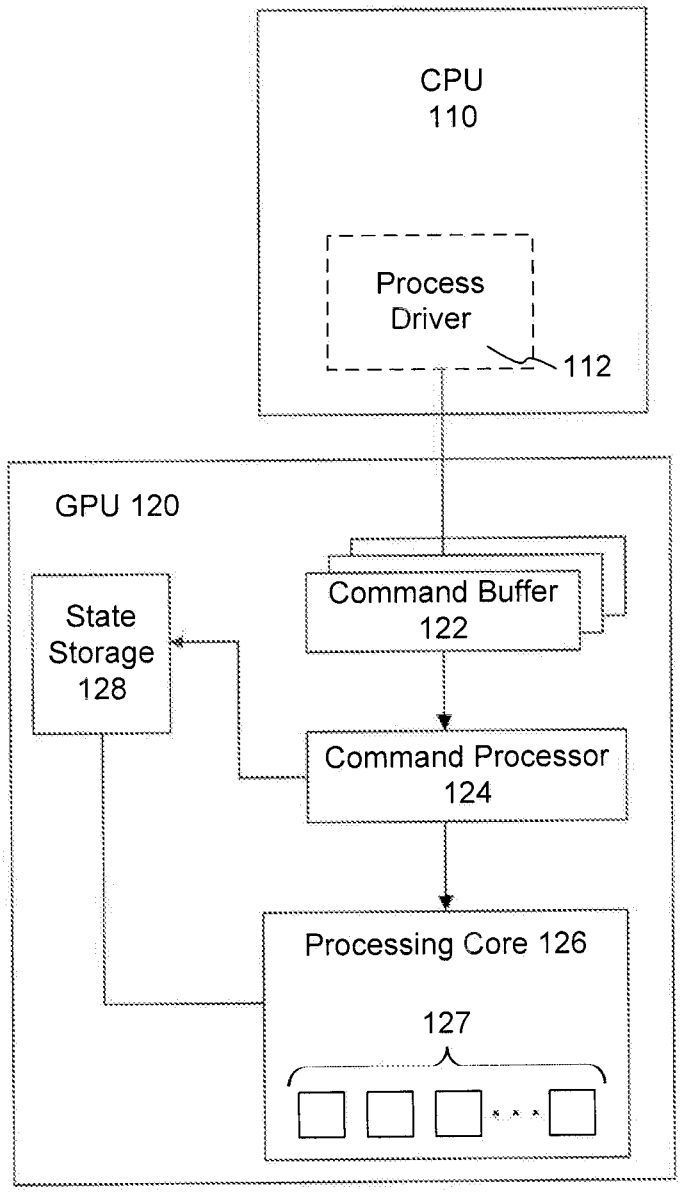


FIG. 1
Conventional

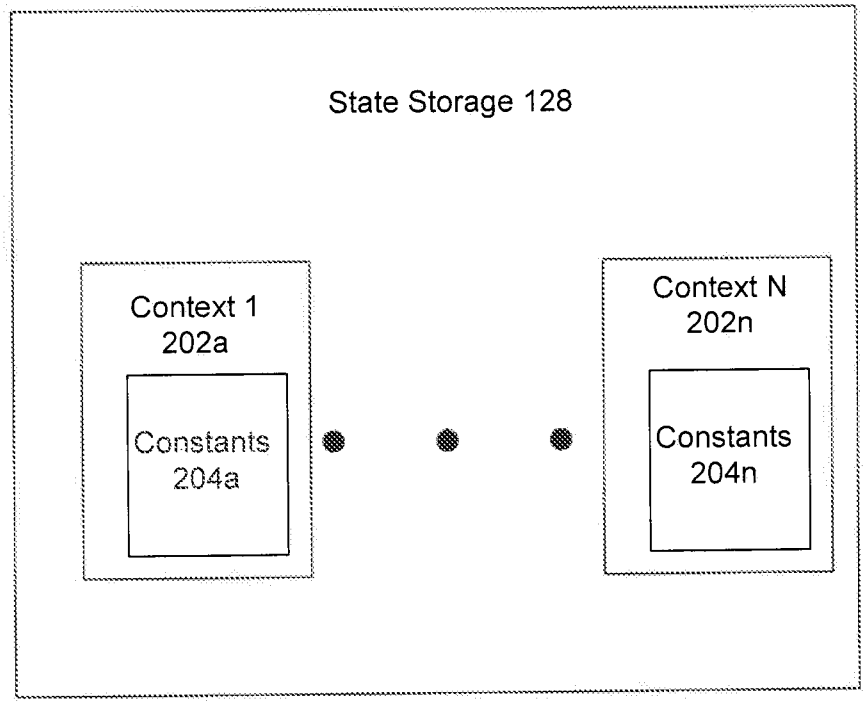


FIG. 2
Conventional

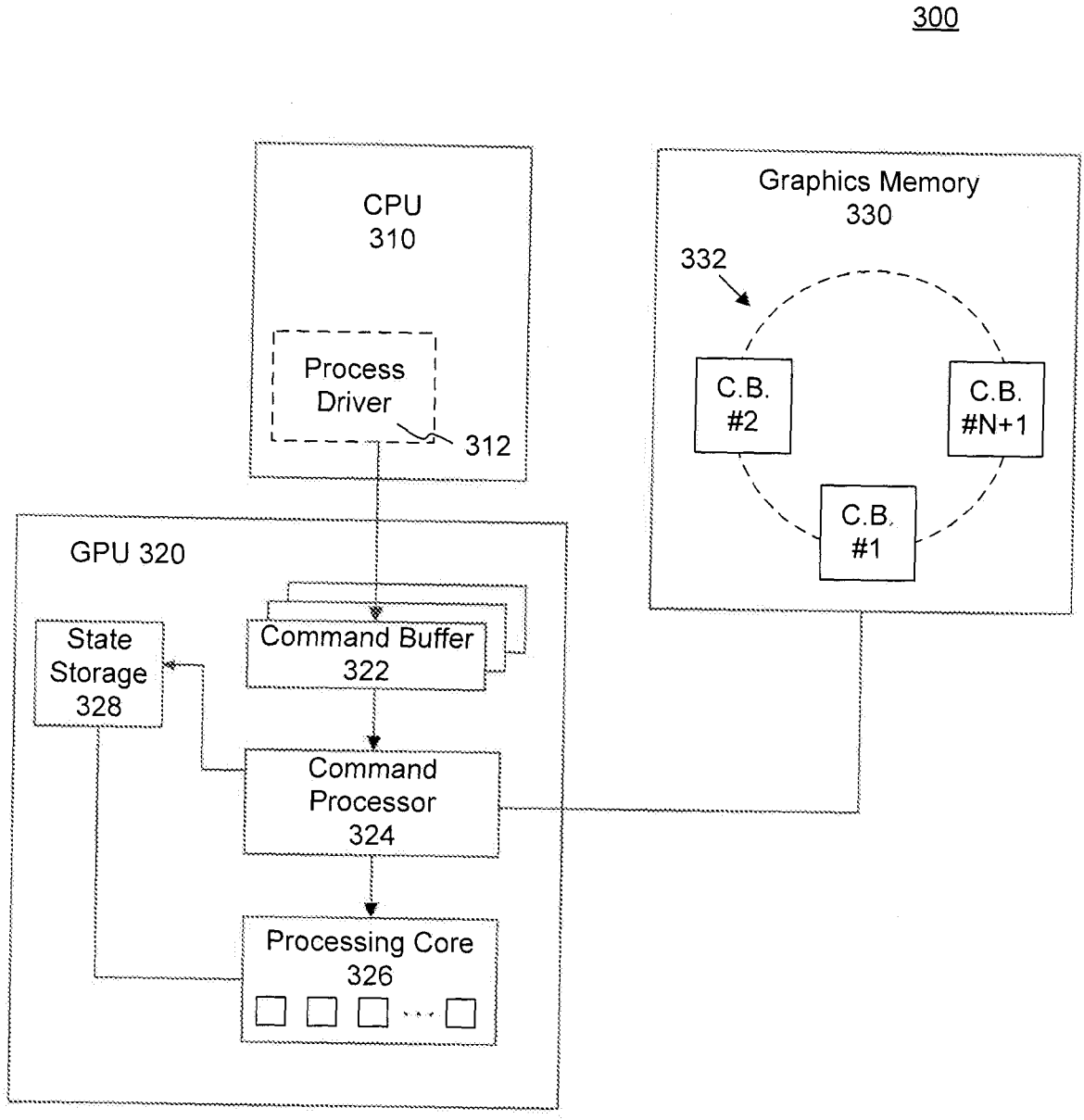

300
CPU
310
Process
Driver
312
Graphics Memory
330
332
C.B.
#2
C.B.
#N+1
C.B.
#1
GPU 320
State
Storage
328
Command Buffer
322
Command
Processor
324
Processing Core
326

FIG. 3

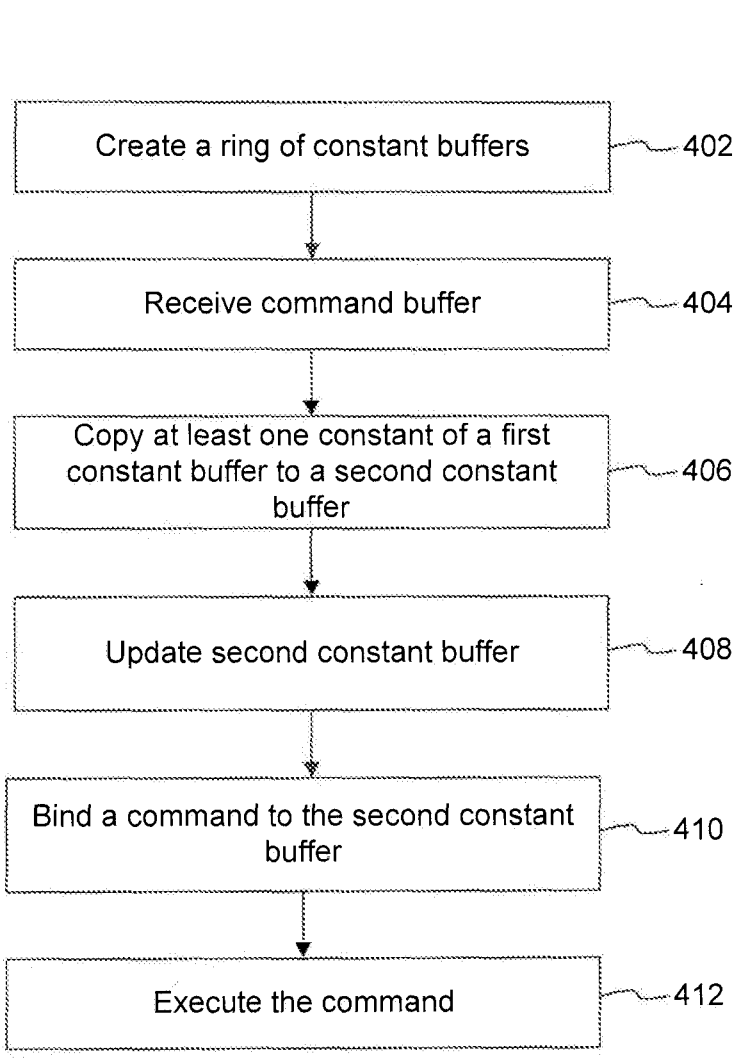

400
Create a ring of constant buffers
402
Receive command buffer
404
Copy at least one constant of a first constant buffer to a second constant buffer
406
Update second constant buffer
408
Bind a command to the second constant buffer
410
Execute the command
412

FIG. 4

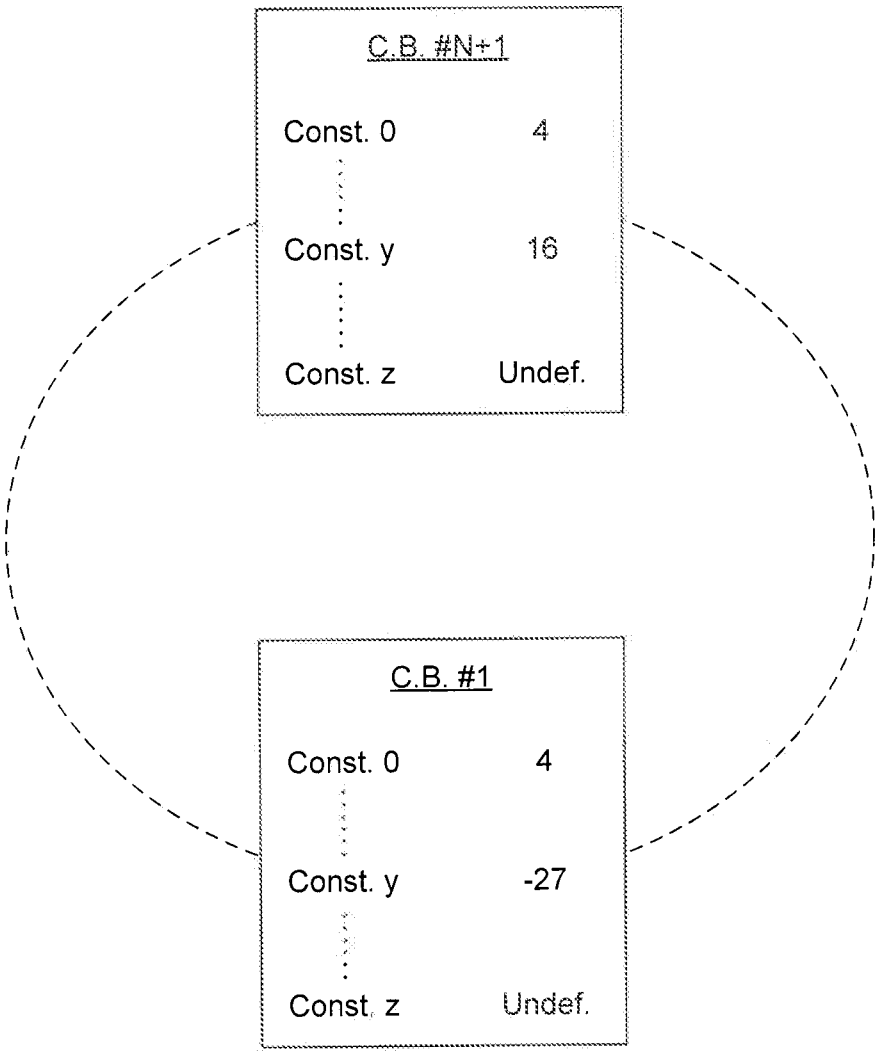

500
C.B. #N+1
Const. 0
4
Const. y
16
Const. z
Undef.
C.B. #1
Const. 0
4
Const. y
-27
Const. z
Undef.

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/069249

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06T1/20
ADD.

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

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal , WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	TAGLIENTI C: "Thread-safe circular queue: tackling the bounded buffer problem", C/C++ USERS JOURNAL, R&D PUBLICATIONS, LAWRENCE, KS, US, vol. 22, no. 6, 1 June 2004 (2004-06-01), pages 42-45, XP009167194, page 44, right-hand column, paragraph 6 - page 45, left-hand column, last paragraph; figures 3,4	1-19

-/--

[X] 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" documentwhich may throw doubts on priority claim(s) orwhich 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	Date of mailing of the international search report
22 February 2013	07/03/2013

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Pierfederici, A

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/069249

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PINHEIRO ET AL.: "Introduction to Multithreaded Rendering and the Usage of Deferred Context in DirectX 11", SBS - PROCEEDINGS OF SBGAMES 2011, vol. TB-02820-001_vOl, 9 November 2011 (2011-11-09), pages 1-5, XP002692613, Salvador, Bahia, Brazil page 2, left-hand column, paragraph 2 - page 2, right-hand column, paragraph 3	1-19
A	CHUNG-WEN HUANG ET AL: "Support of Paged Register Files for Improving Context Switching on Embedded Processors", COMPUTATIONAL SCIENCE AND ENGINEERING, 2009. CSE '09. INTERNATIONAL CONFERENCE ON, IEEE, PISCATAWAY, NJ, USA, 29 August 2009 (2009-08-29), pages 352-357, XP031544528, page 353, right-hand column, paragraph 2	5,6, 14, 15
A	Anonymous: "Microsoft DirectX 10: The Next-Generation Graphics API", November 2006 (2006-11), nvidia, Santa Clara, CA 95050, USA, XP002692614, page 15, paragraph 2 - paragraph 5; figure 7	1-19
A	WEI-CHUN KU ET AL: "VisoMT: A Collaborative Multithreading Multicore Processor for Multimedia Applications with a Fast Data Switching Mechanism", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE SERVICE CENTER, PISCATAWAY, NJ, US, vol. 19, no. 11, 1 November 2009 (2009-11-01), pages 1633-1645, XP011321292, the whole document	1-19